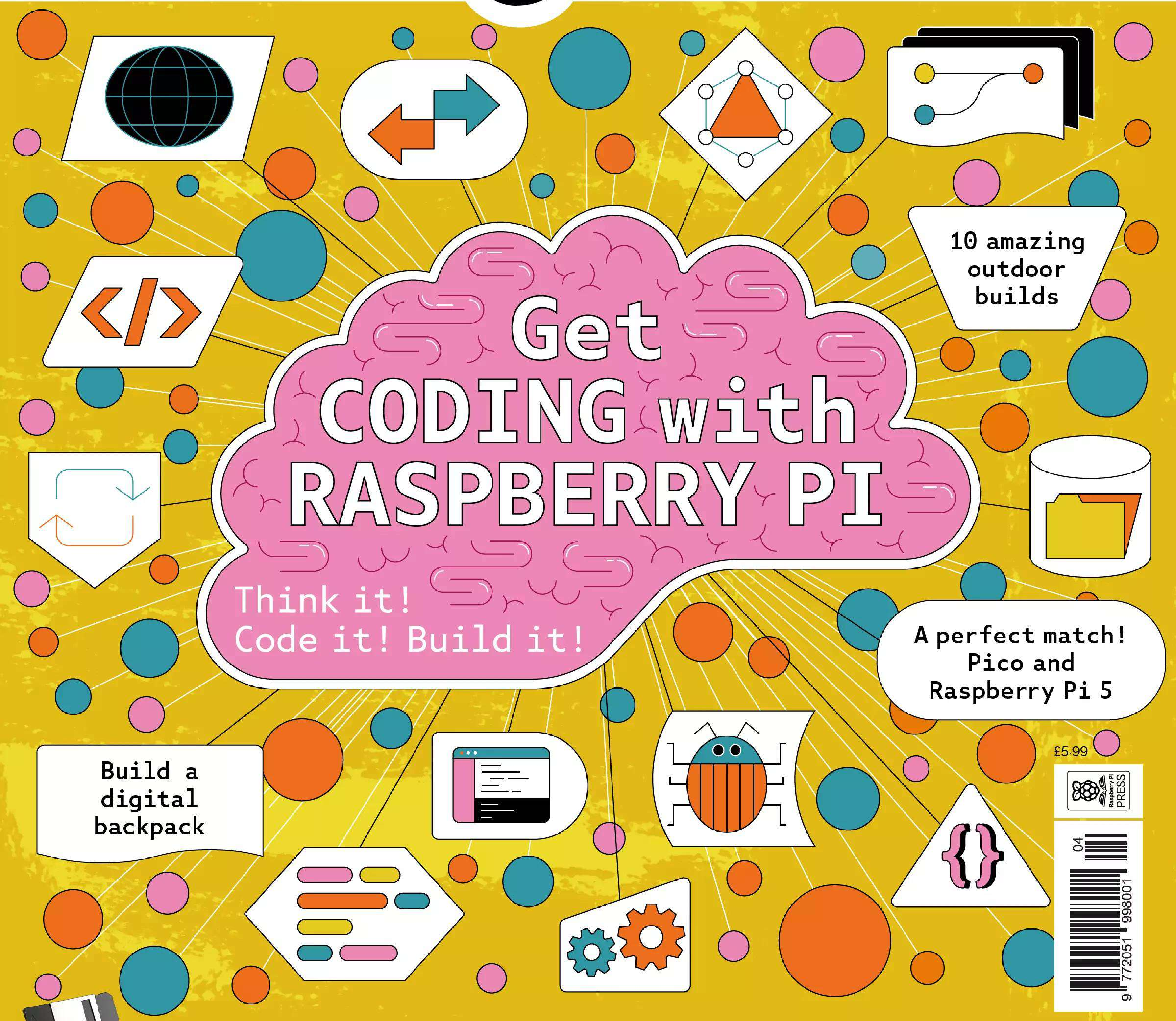
YOUR OFFICIAL RASPBERRY PI MAGAZINE

The MagPi

Issue 140 | April 2024 | magpi.cc

The official Raspberry Pi magazine



Get CODING with RASPBERRY PI

Think it! Code it! Build it!

10 amazing outdoor builds

A perfect match! Pico and Raspberry Pi 5

Build a digital backpack

£5.99

Raspberry Pi PRESS

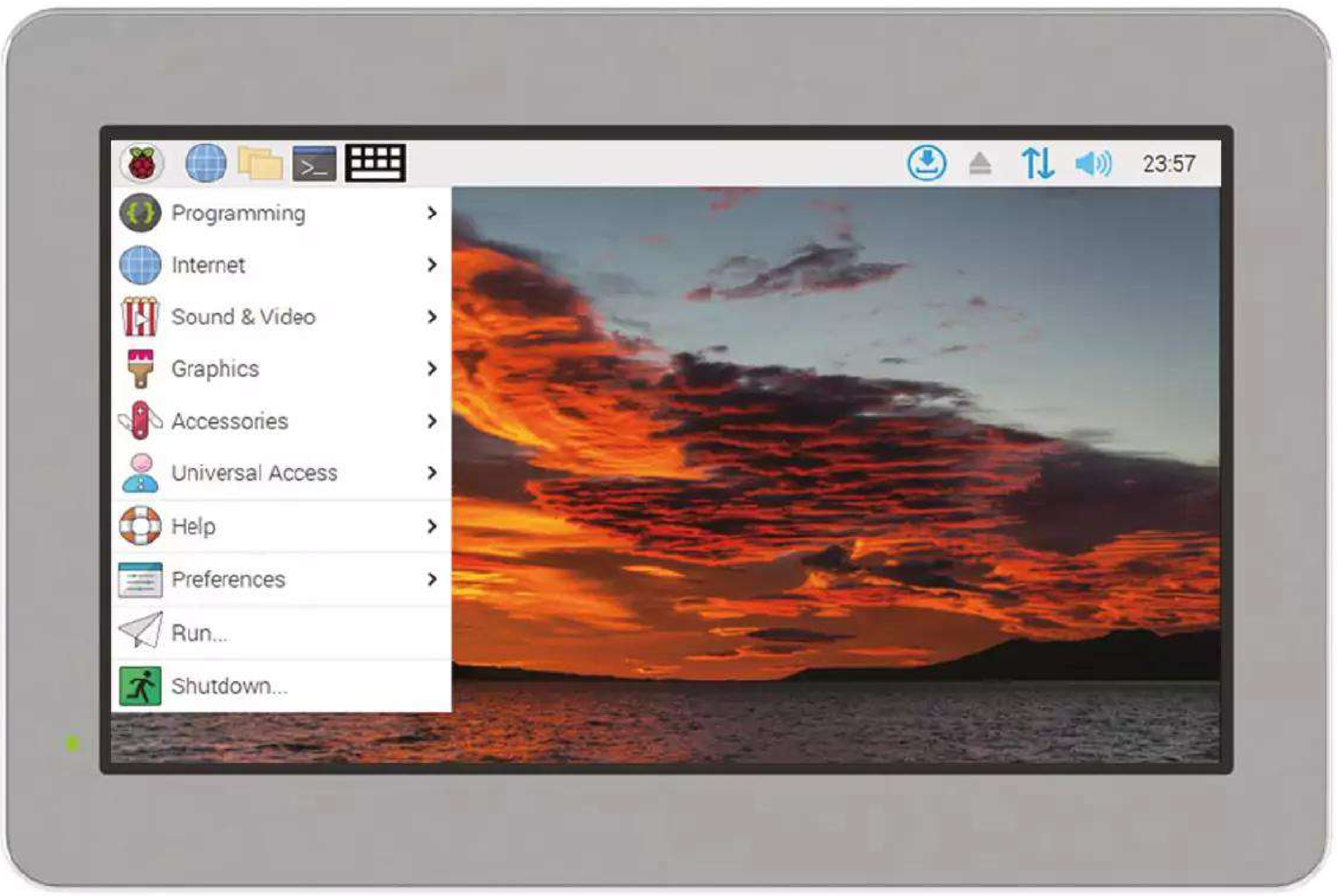
04

9 772051 998001

SAVE YOUR SOFTWARE! RETRO FLOPPY DISK & CD GUIDE

Industrial Raspberry Pi

ComfilePi



UL LISTED

Powered by Raspberry Pi

The ComfilePi is a touch panel PC designed with high-tolerant components and no moving parts for industrial applications. It features a water-resistant front panel, touchscreen, color LCD (available in various sizes), RS-232, RS-485, Ethernet, USB, I2C, SPI, digital IO, battery-backed RTC (real-time clock), and piezo buzzer.

Use the rear-panel 40-pin GPIO header to expand its features and capabilities with additional I/O boards. The ComfilePi is UL Listed and employs Raspberry Pi Compute Module.

Visit www.comfiletech.com

© copyright COMFILE Technology, Inc. ALL RIGHTS RESERVED

COMFILE TECHNOLOGY

WELCOME
to The MagPi 140

This month is special for me. This month we cover learning to code with Raspberry Pi (page 34). Teaching kids of all ages how to code is what Raspberry Pi, and by extension, *The MagPi* exists for.

Coding is close to my heart, ever since I was pushed in front of a BBC Micro Model B at school. And I've written a Python tutorial that shows you how to create a to-do list (page 42). It's great to go over all the basics of Python all over again.

If you are already a coding ninja then there's plenty of fun to be found in making things with Raspberry Pi and code. In particular, I really enjoyed KG's exploration of using floppy disk drives and CD-ROM drives with Raspberry Pi (page 56) and Rosie's write-up of a mini observatory based on Raspberry Pi (page 20).

Making stuff work with code is the most fun you can have with computers. This month is packed with ideas!

Lucy Hattersley Editor

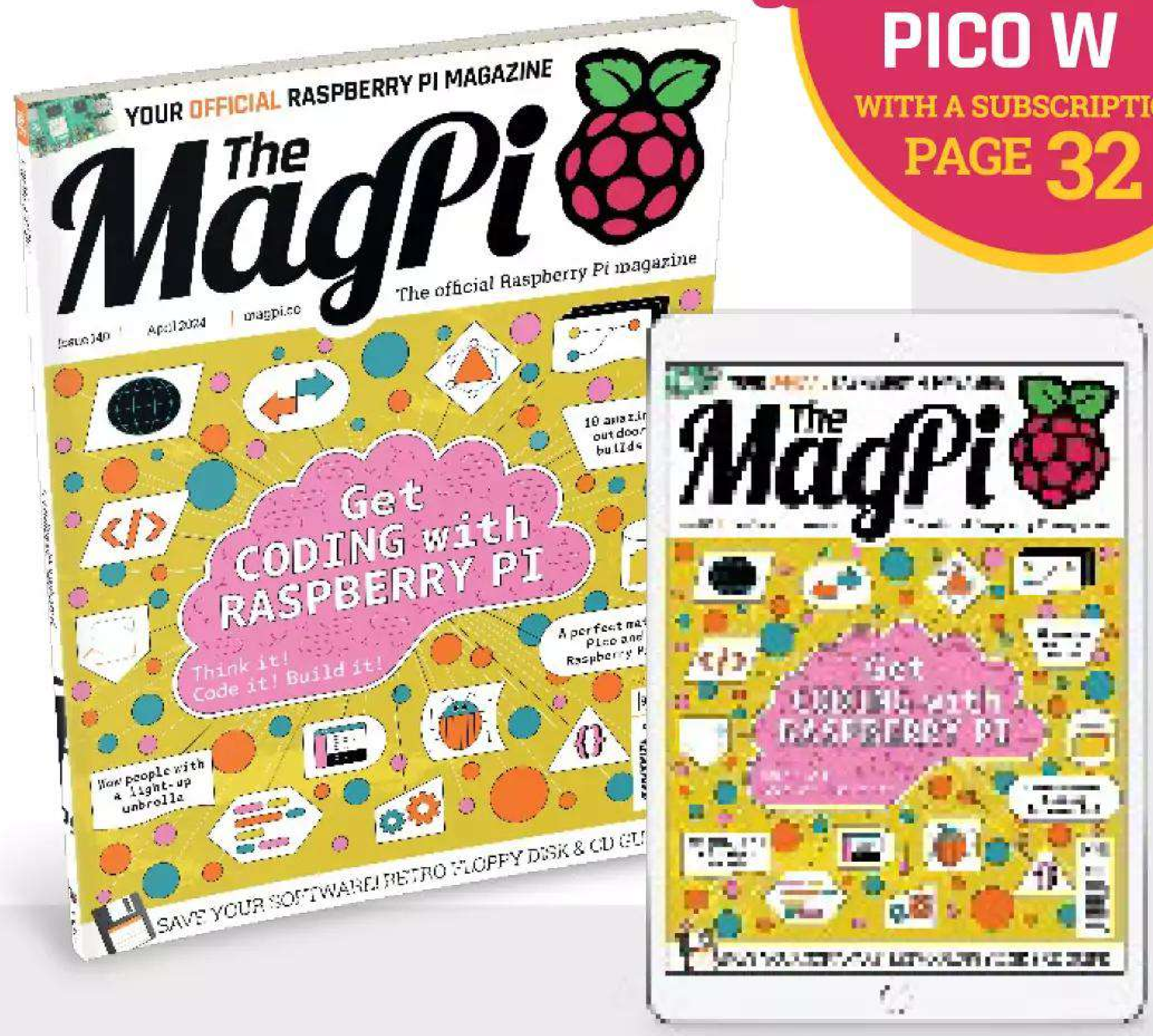
EDITOR

Lucy Hattersley

Lucy is editor of *The MagPi* and this month she has a new nose for sniffing out the best tech projects.

magpi.cc

GET A
RASPBERRY PI
PICO W
WITH A SUBSCRIPTION!
PAGE 32


YOUR OFFICIAL RASPBERRY PI MAGAZINE
The MagPi
The official Raspberry Pi magazine
Issue 140 | April 2024 | magpi.cc
Get CODING with RASPBERRY PI
Think it! Code it! Build it!
10 amazing outdoor builds
A perfect match Pico and Raspberry Pi
Wow people with a light-up umbrella
SAVE YOUR SOFTWARE! RETRO FLOPPY DISK & CD GU

New look

Same focus

We’ve refreshed our brand, but our commitment to customer-centric experiences remains constant.

And as always, our goal is to accelerate progress for every designer, buyer, and builder.

Learn more at digikey.co.uk

DigiKey

we get technical

DigiKey is a franchised distributor for all supplier partners. New products added daily. DigiKey and DigiKey Electronics are registered trademarks of DigiKey Electronics in the U.S. and other countries. © 2024 DigiKey Electronics, 701 Brooks Ave. South, Thief River Falls, MN 56701, USA

ECIA MEMBER
Supporting The Authorized Channel

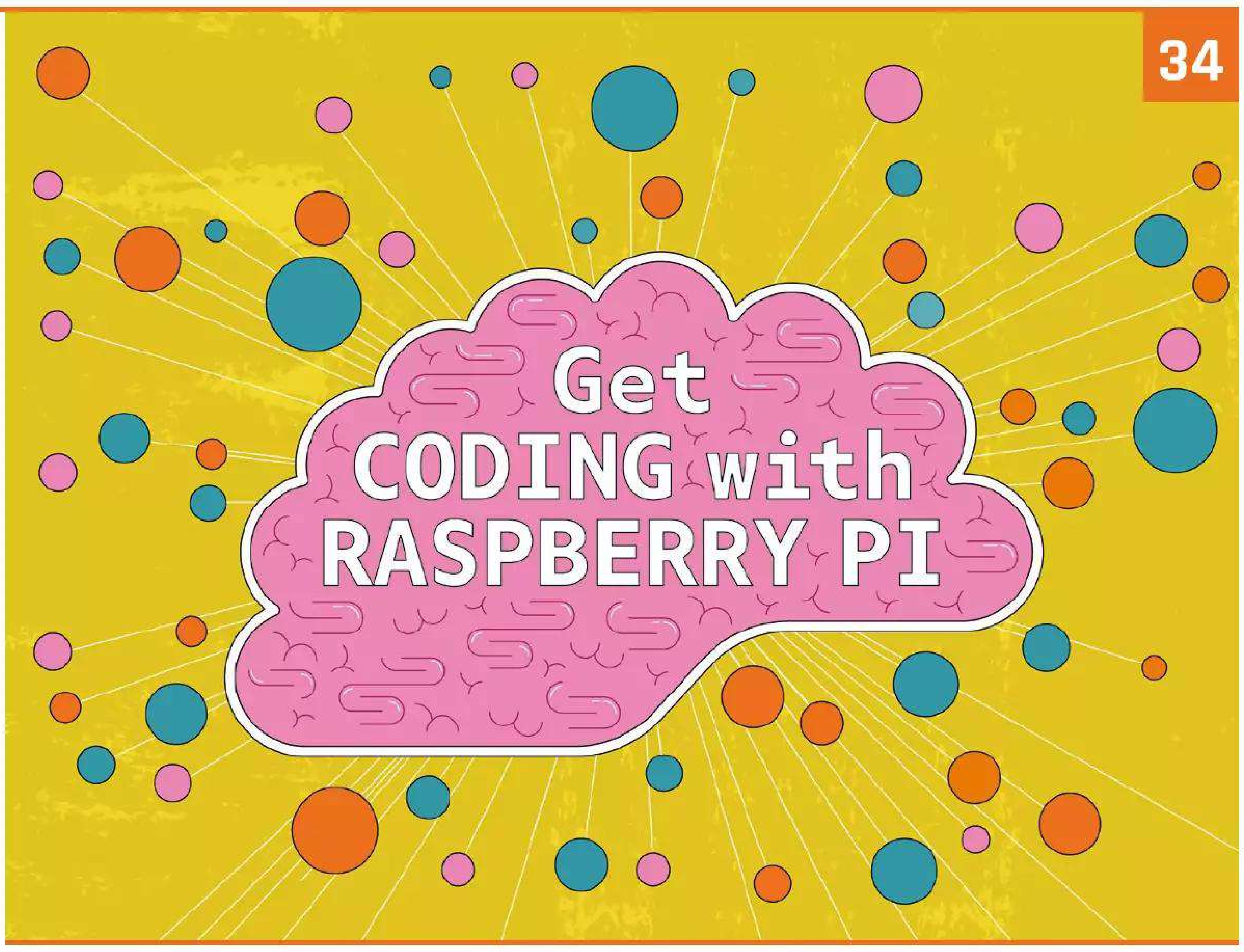
Contents

Issue 140 April 2024

Cover Feature

Regulars

Project Showcases



Red telephone

Mini observatory

The MagPi is published monthly by Raspberry Pi Ltd, 194 Cambridge Science Park, Milton Road, Cambridge, England, CB4 0AB. Publishers Service Associates, 2406 Reach Road, Williamsport, PA, 17701, is the mailing agent for copies distributed in the US and Canada. Application to mail at Periodicals prices is pending at Williamsport, PA. POSTMASTER: Send address changes to The MagPi, c/o Publishers Service Associates, 2406 Reach Road, Williamsport, PA, 17701.

Tutorials

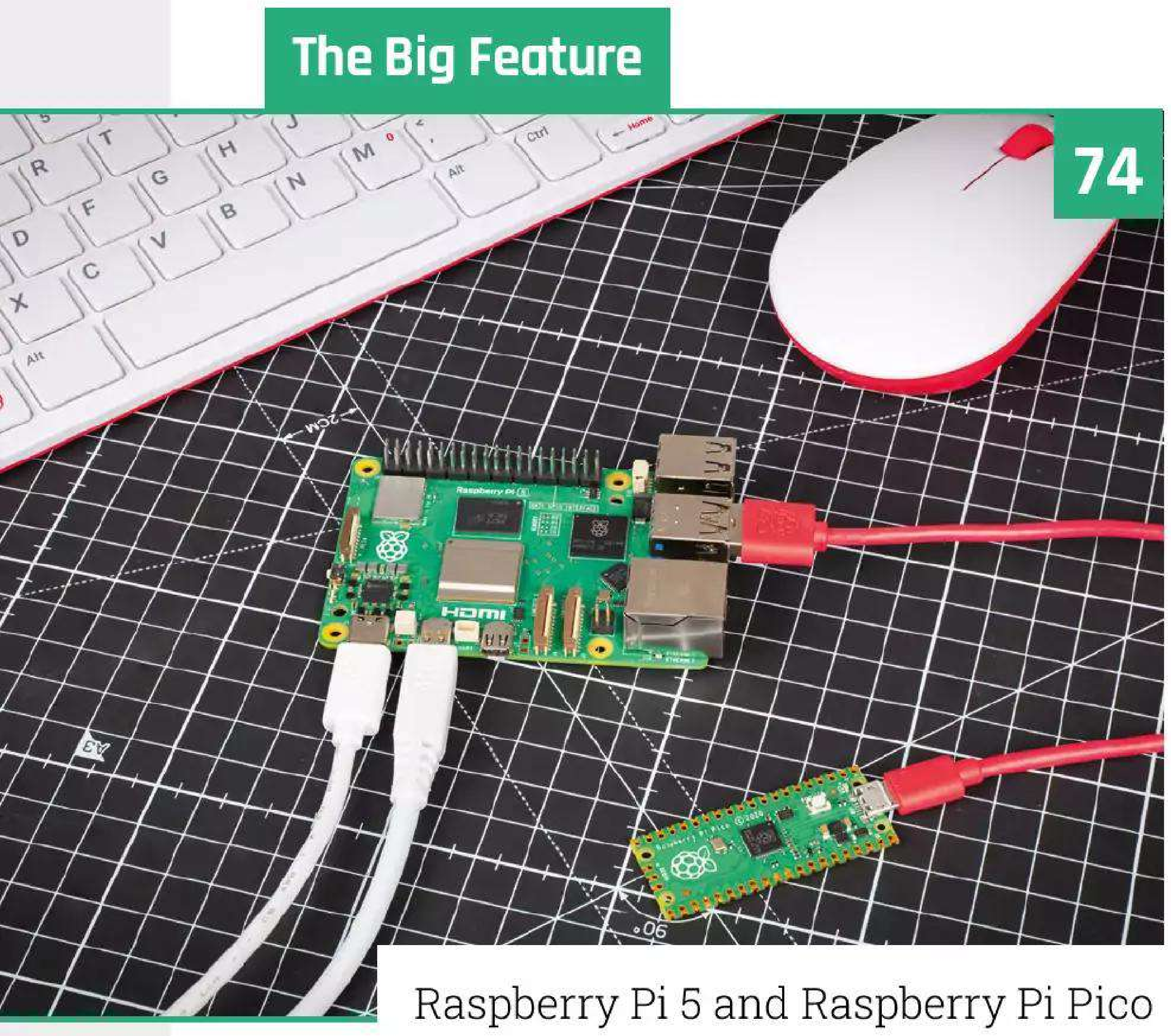
56

Add CD and floppy drives to Raspberry Pi 5

62

Upcycling with Sonos – part 2

The Big Feature



74

Raspberry Pi 5 and Raspberry Pi Pico

80

Passive Cooling Open CNC Case

Reviews

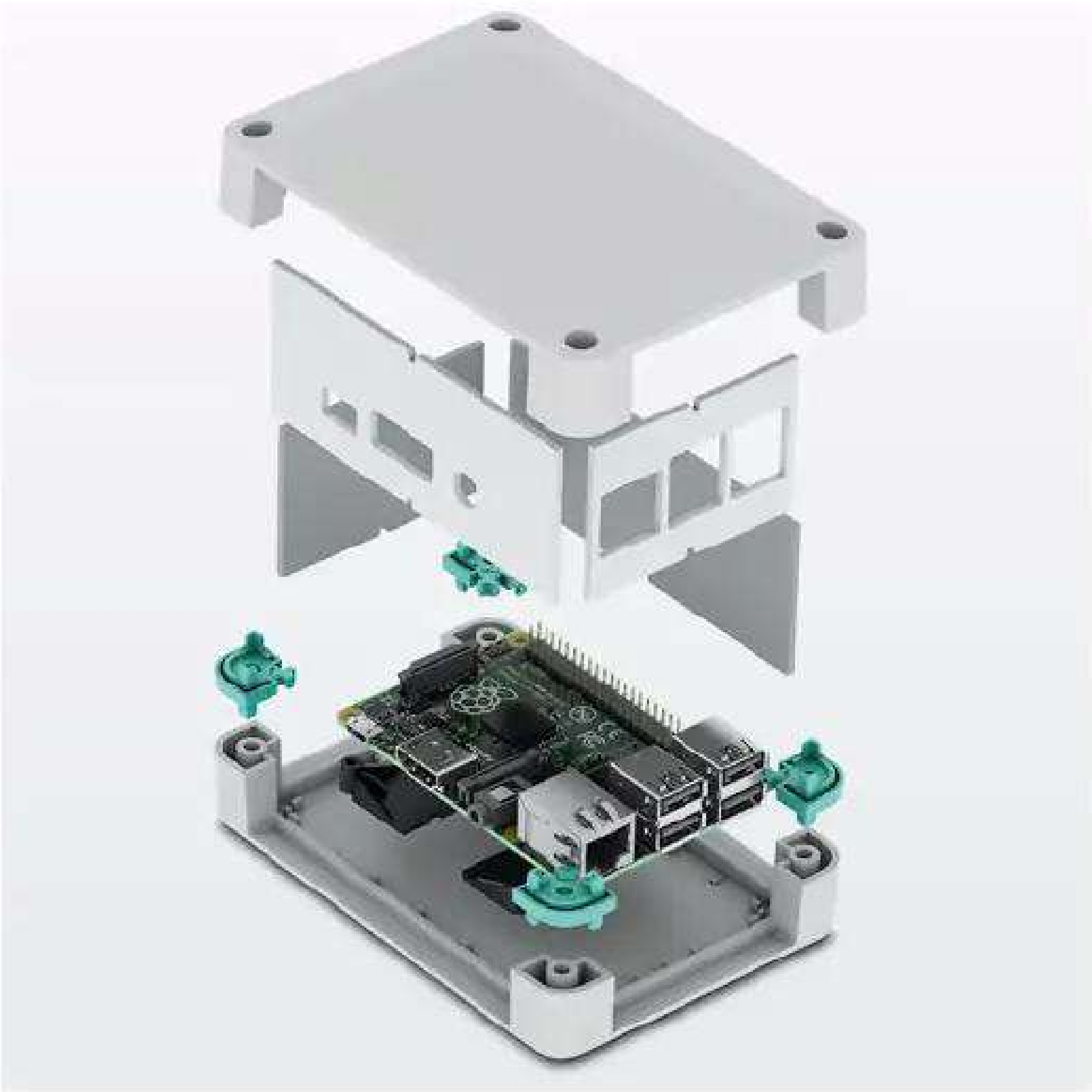
Community

86

Interview: Sara Parodi

WIN 1 OF 5 RASPBERRY JAM **BUNDLES** 96

DISCLAIMER: Some of the tools and techniques shown in The MagPi magazine are dangerous unless used with skill, experience, and appropriate personal protection equipment. While we attempt to guide the reader, ultimately you are responsible for your own safety and understanding the limits of yourself and your equipment. Children should be supervised. Raspberry Pi Ltd does not accept responsibility for any injuries, damage to equipment, or costs incurred from projects, tutorials or suggestions in The MagPi magazine. Laws and regulations covering many of the topics in The MagPi magazine are different between countries, and are always subject to change. You are responsible for understanding the requirements in your jurisdiction and ensuring that you comply with them. Some manufacturers place limits on the use of their hardware which some projects or suggestions in The MagPi magazine may go beyond. It is your responsibility to understand the manufacturer's limits.



Raspberry Pi finds a new home

Versatile enclosure for Raspberry Pi B+

The new UCS Universal Case System is now available with ready made cut-outs for the 7” touch display and standard connections of the Raspberry Pi B+ single board computers. The 237 x 195 x 47mm housings are available in black or grey and are suitable for wall or desktop mounting

For additional information call 01952 681700 or visit

https://phoe.co/ucs-series

PHOENIX CONTACT

INSPIRING INNOVATIONS

Sandeep Mistry profile

A Principal Software Engineer at Arm and creator of excellent Raspberry Pi projects. By **Rob Zwetsloot**

Over the years we've found that however senior you are as a software or hardware engineer, tinkering with a Raspberry Pi or Pico is pretty universal. We've featured projects from engineers at Raspberry Pi in the past but it's true in other companies – like Sandeep Mistry from Arm who works within the IoT Line of Business (LoB). He creates projects that illustrate what folks can do with Raspberry Pi technology.

"I've been making things from a young age." Sandeep tells us. "It started off with building things like LEGO, then moving onto higher tech things after getting the opportunity to attend the Virtual Ventures summer camp (**carleton.ca/vv**). I was very fortunate that the high school I went to offered courses in computer science, so I had the opportunity to learn about how to make software applications before going to university. After getting a university degree in computer systems engineering, I've worked as a software engineer creating software for end products and platforms."

After working for various companies as a software engineer in Ottawa, and also working remotely with Arduino, he started looking for a new way to use his expertise with embedded and mobile devices. There's not much better place than Arm for that, and when an acquaintance of his mentioned a position coming up, he jumped at the chance.

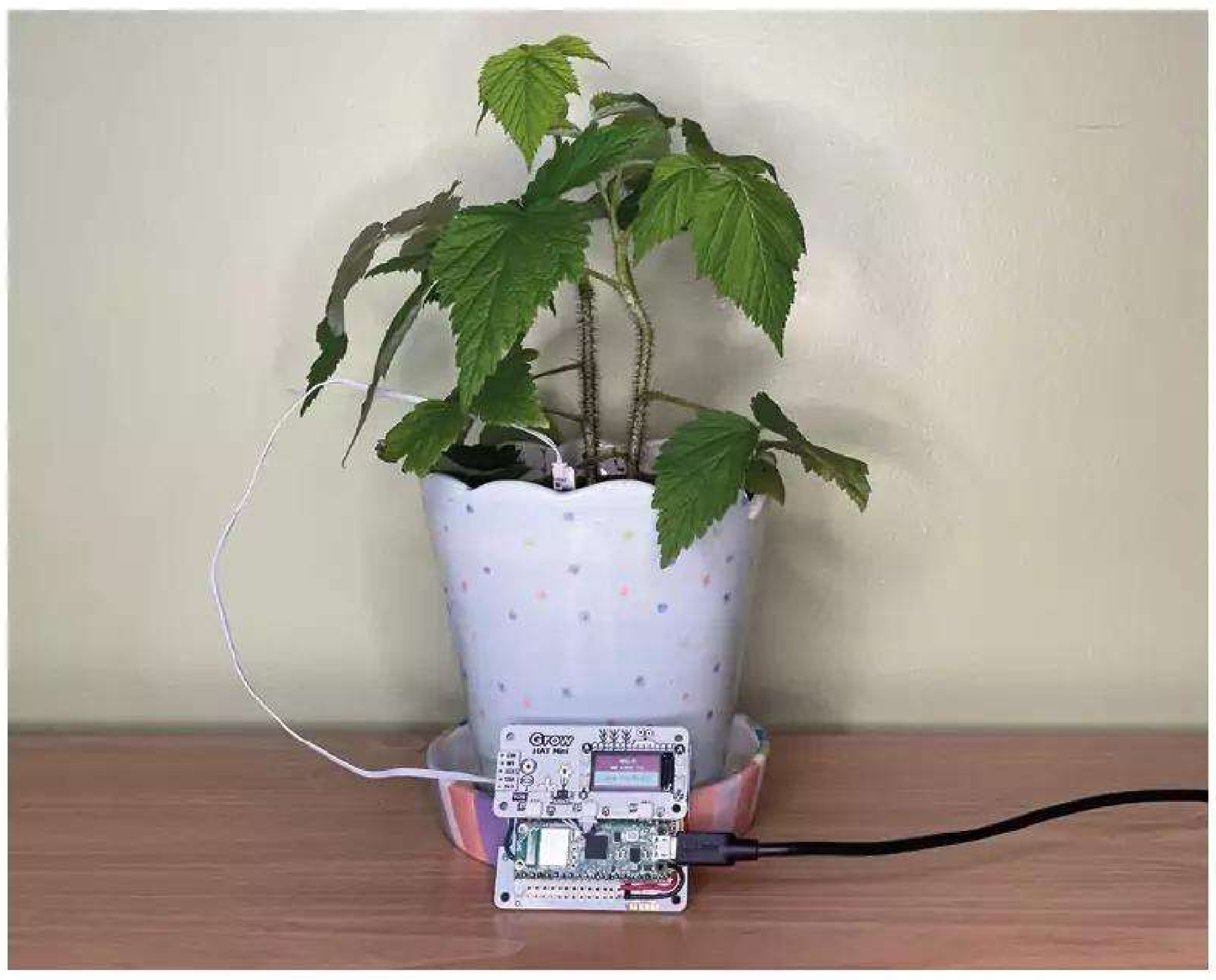
He'd already worked on a Raspberry Pi Pico project before joining Arm – if you cast your mind back to 2021 around the time Pico was released, he built a way to add Ethernet to Pico via the PIO. It was even featured on the blog. Since then, he's been putting out software and hardware projects of varying levels of complexity, and has really shown off the power of Raspberry Pi in the process.

Sandeep's builds

Give a plant a personality

"[A favourite project of mine] was using Pico W to create a plant that texts you when it needs water, after it's been watered, good morning/night, and random jokes," Sandeep says. "It was the first time I used MicroPython, and it was fun to use the Pico W to give a plant a personality."

▶ **magpi.cc/picoplant**

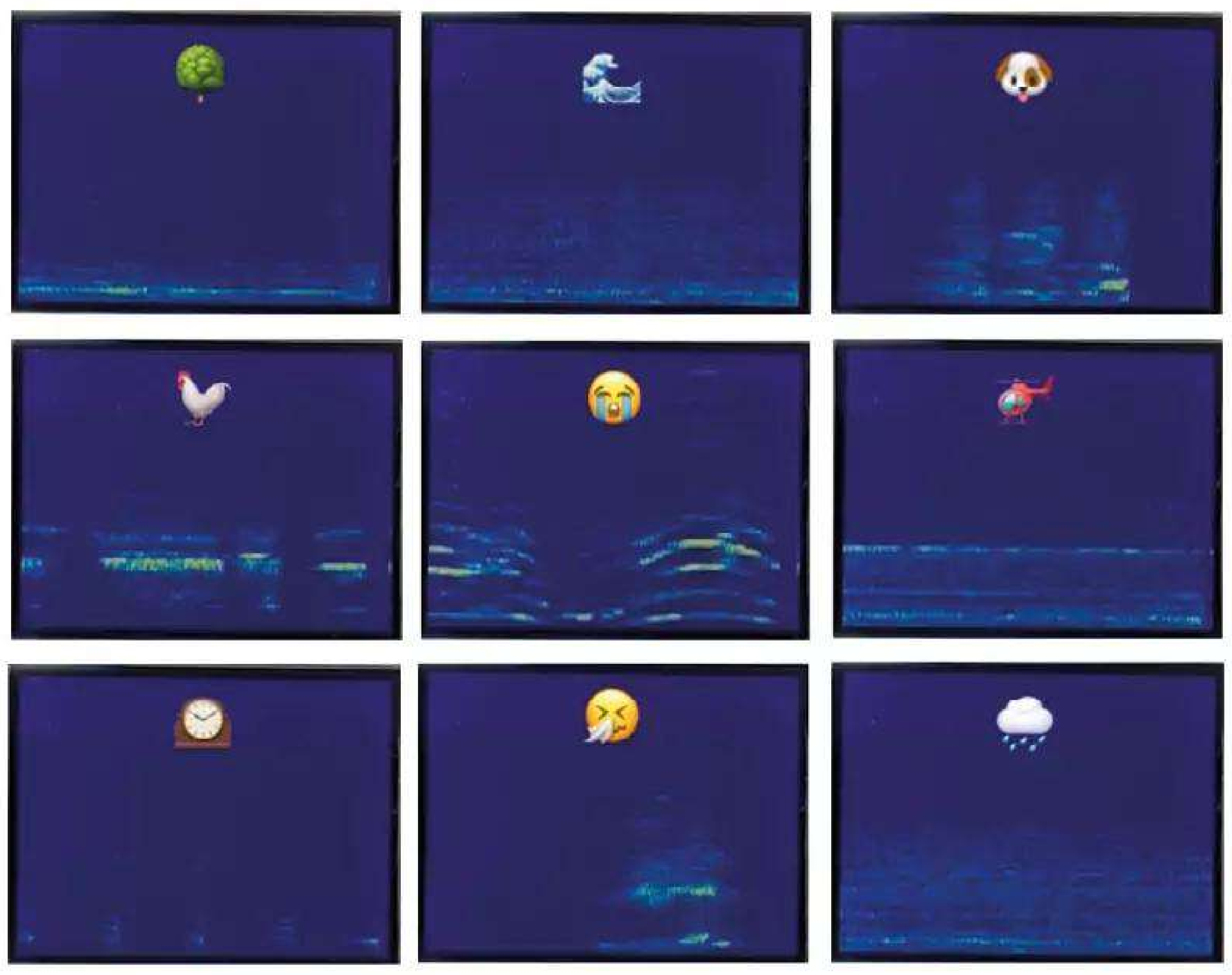


▲ Now your plant can text you jokes!

See sound in real time

Building on the Pico microphone project, this project shows how you can create cool visualisers for sound using a display also hooked up to Pico. These specific visualisations are audio spectrograms, which display sound as amplitude over time for cool visual effects.

▶ **magpi.cc/picosoundviz**



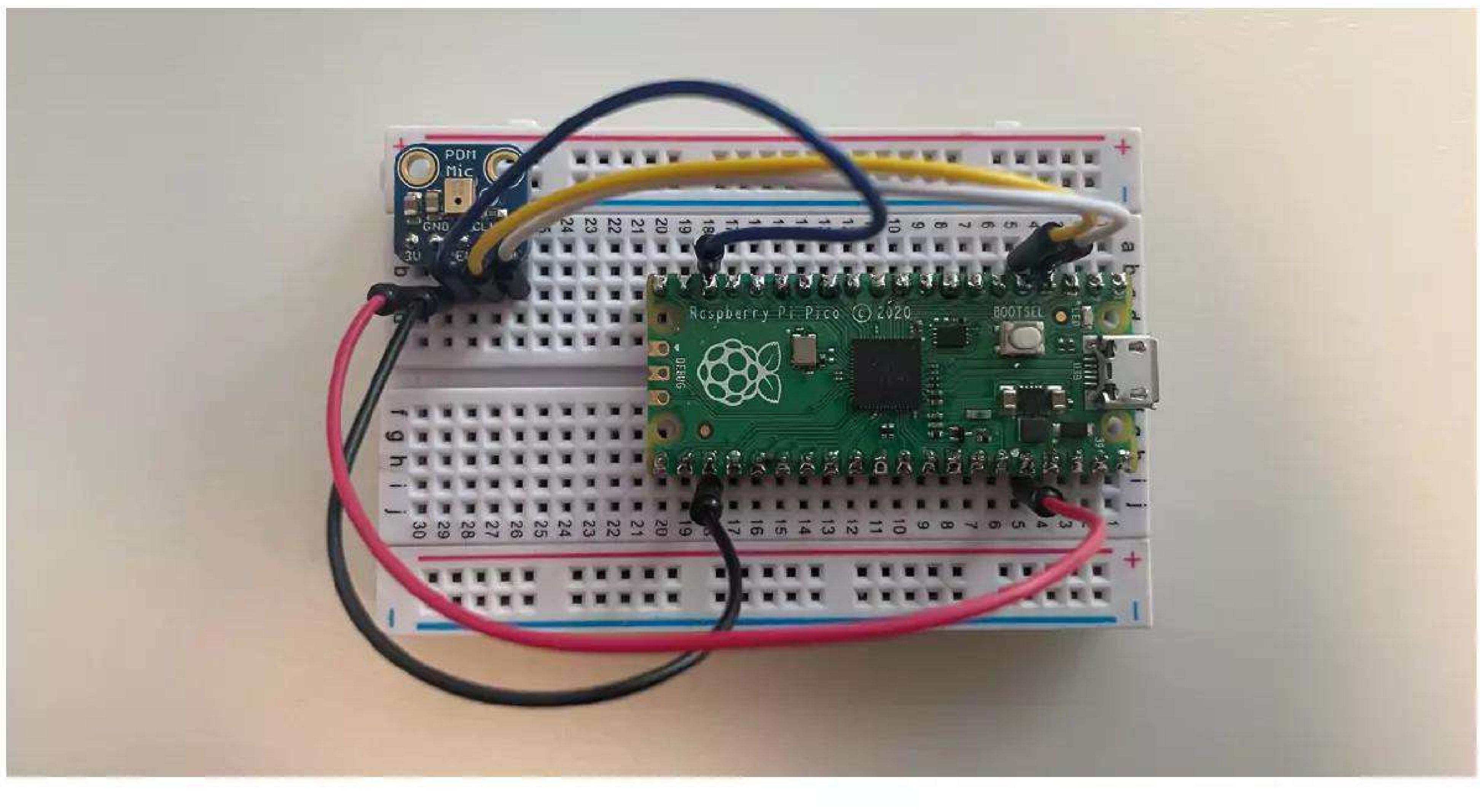
▲ Emoji representations of different sound types – they look like ripples in puddles

Create a USB microphone

Pico can do a lot with its PIO capabilities – and it also has an ADC (analogue-to-digital converter) onboard. Combining the ability to connect to a system as a USB device while also listening to a microphone means you can create a custom Pico-powered mic. It's an interesting and fairly cheap way to add a mic to a computer.

▶ **magpi.cc/picomic**

▼ This simple build uses a PDM mic for sound input



Backpack Cyberdeck

An easily transportable Raspberry Pi computer offers lots of customisation options, reasoned one maker. **Rosie Hattersley** packs up

MAKER

Davide Marchetti

Metal industry professional and tech experimenter Davide loves being able to take his mobile lab from place to place – on two wheels

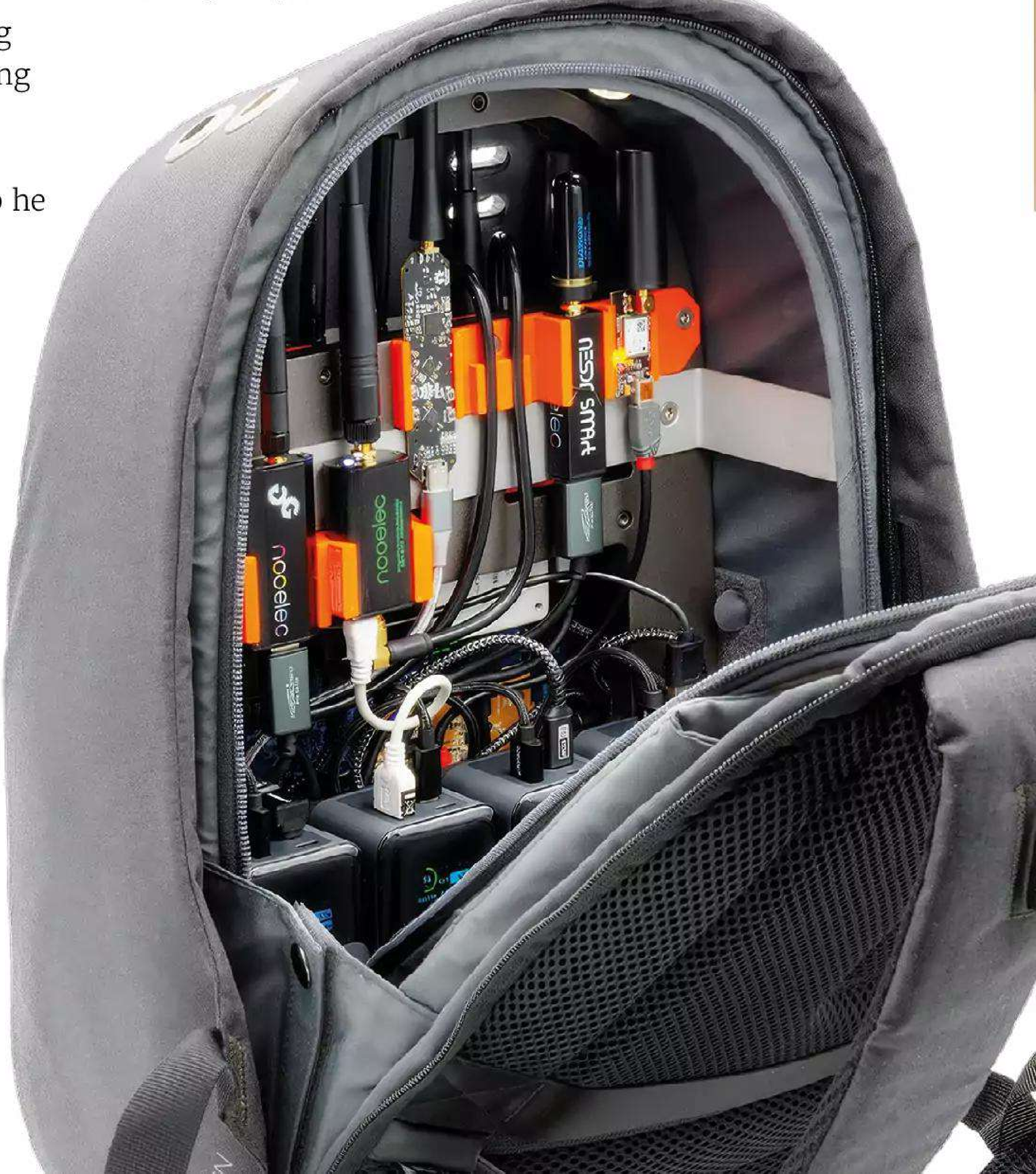
bagbuilds.com

Like many readers of *The MagPi*, an early fascination with computers led maker Davide Marchetti to teach himself as much as he could about programming and electronics. His fascination with tech was enhanced through hands-on projects such as creating a hydroponic greenhouse system and building an Arduino-controlled pantograph.

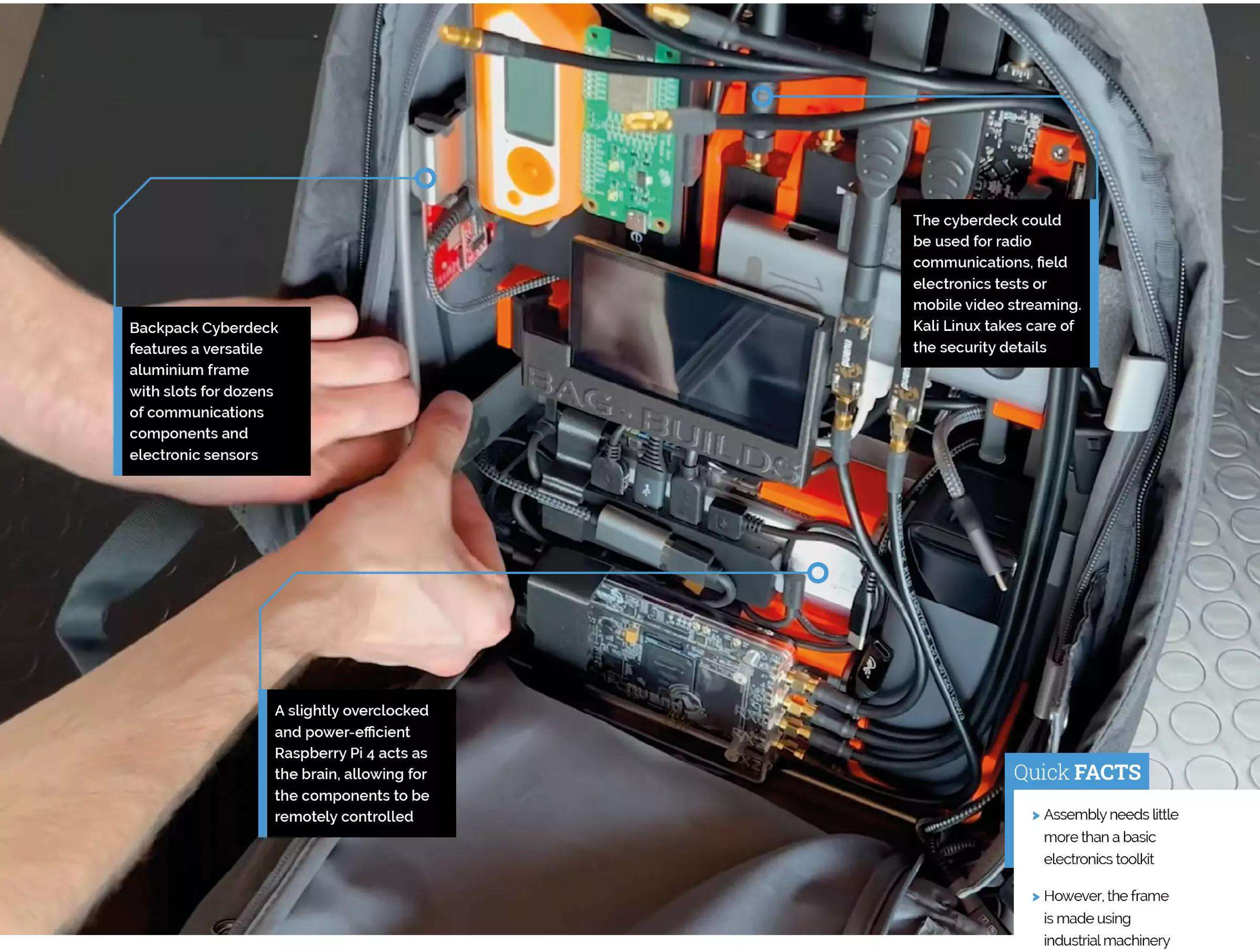
Davide recently embarked on projects covering 3D printing and wireless communications, leading him to explore the wonderful possibilities of Raspberry Pi. The Backpack Cyberdeck is a self-contained, Raspberry Pi 4-powered, portable lab he can take anywhere with him on his bike. "Even when I travel, I want all my stuff with me for all occasions and always ready," he says. It is housed in a fantastic plywood and 3D-printed frame, which is both lightweight and keeps everything securely in place.

A great framework

The Backpack Cyberdck was inspired by Davide's need to move his experiments easily without setup breakdowns – he works in the metalworking industry, and reasoned that a custom-designed frame that fits inside a commercially available backpack would come in mighty useful for other people as well as himself. He says the idea is that hobbyists and professionals can carry, use and interact with a variety of devices on the go. Raspberry Pi 4 was chosen for its compactness and power efficiency and serves as the 'brain' of Davide's mobile setup. It allows him to remotely control devices mounted on the frame. The project runs on open-source software



▼ The surprising contents of an otherwise ordinary-looking backpack



Backpack Cyberdeck features a versatile aluminium frame with slots for dozens of communications components and electronic sensors

The cyberdeck could be used for radio communications, field electronics tests or mobile video streaming. Kali Linux takes care of the security details

A slightly overclocked and power-efficient Raspberry Pi 4 acts as the brain, allowing for the components to be remotely controlled

Quick **FACTS**

- Assembly needs little more than a basic electronics toolkit
- However, the frame is made using industrial machinery
- Including press brakes for metal parts
- But a 3D-printed one could be designed instead
- The backpack is great for mobile tech support

– "primarily GNU Radio for wireless communications analysis, and Kali Linux tools for security and penetration testing tasks". He says the build cost was "moderate, reflecting the price of the Raspberry Pi, the backpack, and some additional electronic components like the RTL-SDR".

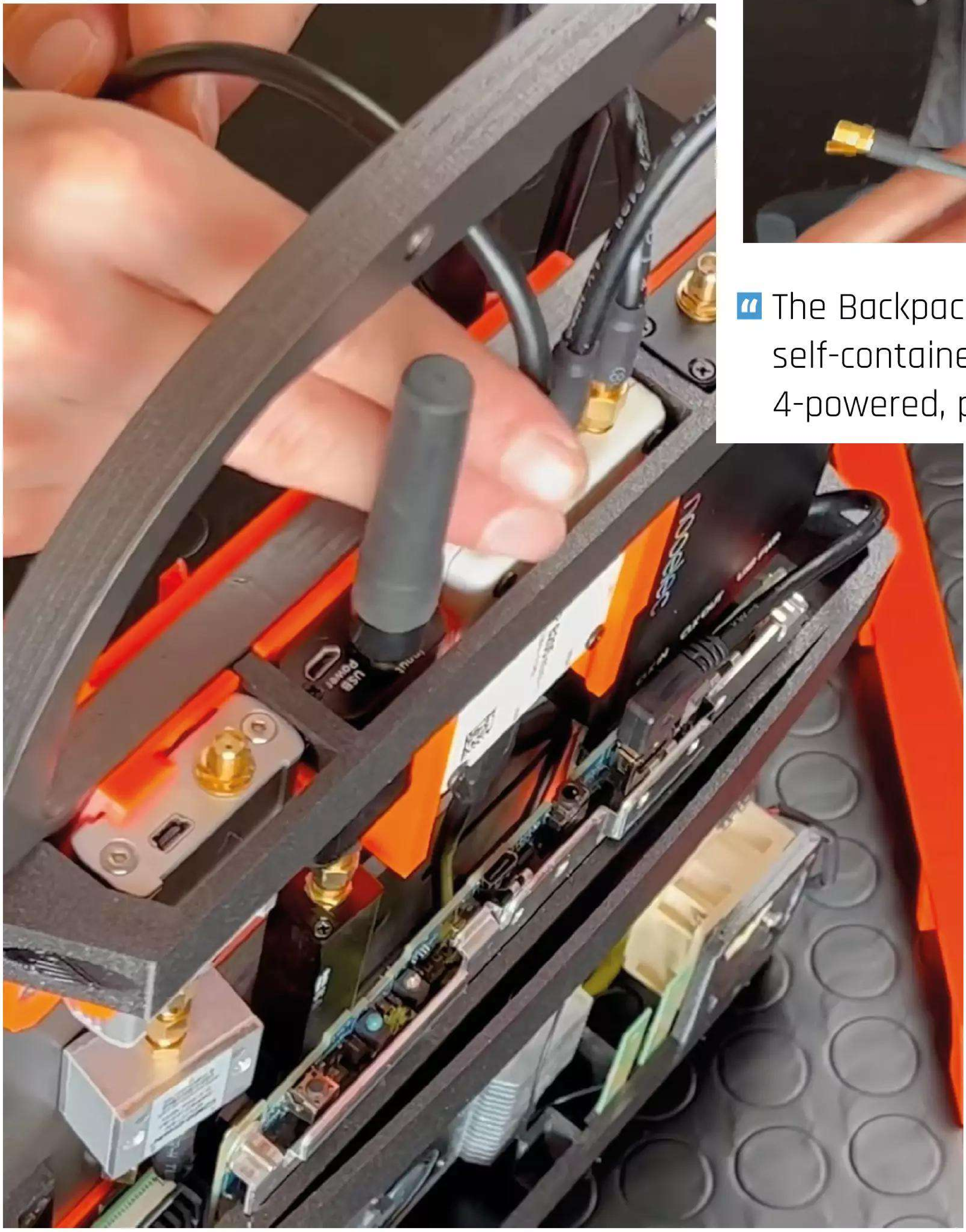
Sharing the build photos on Facebook (**magpi.cc/backpackFB**), Davide explains that it transforms a simple backpack into a customisable platform, allowing for the creation of mobile workstations, entertainment systems, or unique projects through 3D-printed attachments". The Backpack Cyberdeck avoids the risk of damage or subsequent discomfort because there is no desk outdoors. Having cycled to his destination he can conduct his outdoor experiments "the most comfortable way".

Bag your own

For Davide, one advantage of using Raspberry Pi is its compact size, which allowed him to design a system that was both powerful and practical. The design is entirely original, with all parts created or modified by him to suit his project's requirements.

Interest in the backpack led Davide to set up Bag Builds as an online custom bag business, but he designed the Backpack Cyberdeck's frame for DIY enthusiasts with a passion for 3D printing. He shares STL files for several versions of the Backpack Cyberdeck on his GitHub page (**magpi.cc/bagbuildsgit**), with customisation options for the hardware components all designed to fit neatly into a standard backpack. There is plenty of opportunity for DIY builds to be customised, since some users

The robust frame acts as a rack, making the Backpack Cyberdeck extremely adaptable



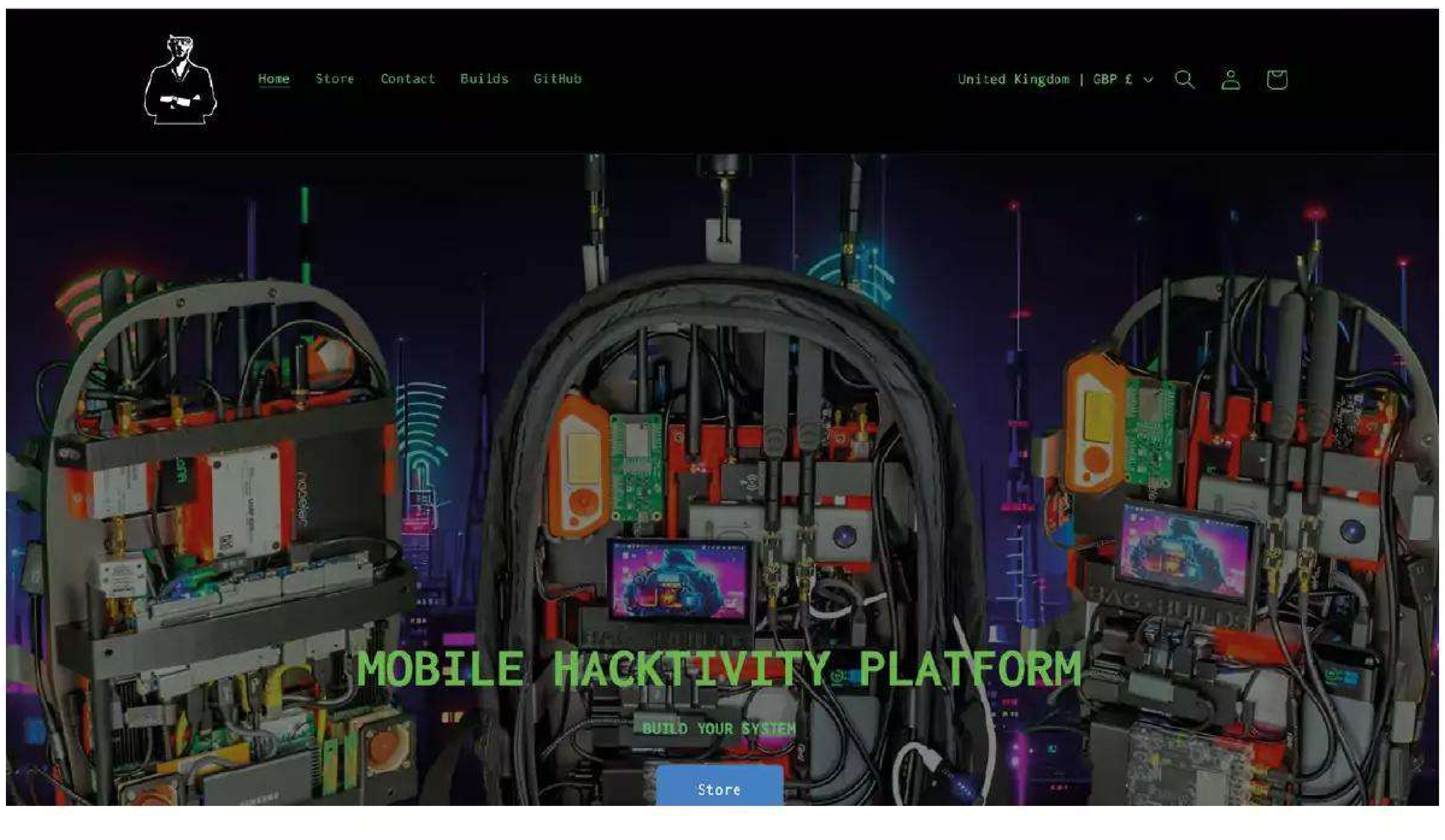
> “The Backpack Cyberdeck is a self-contained, Raspberry Pi 4-powered, portable lab”

may need holes to allow antennae through, and to fit cables.

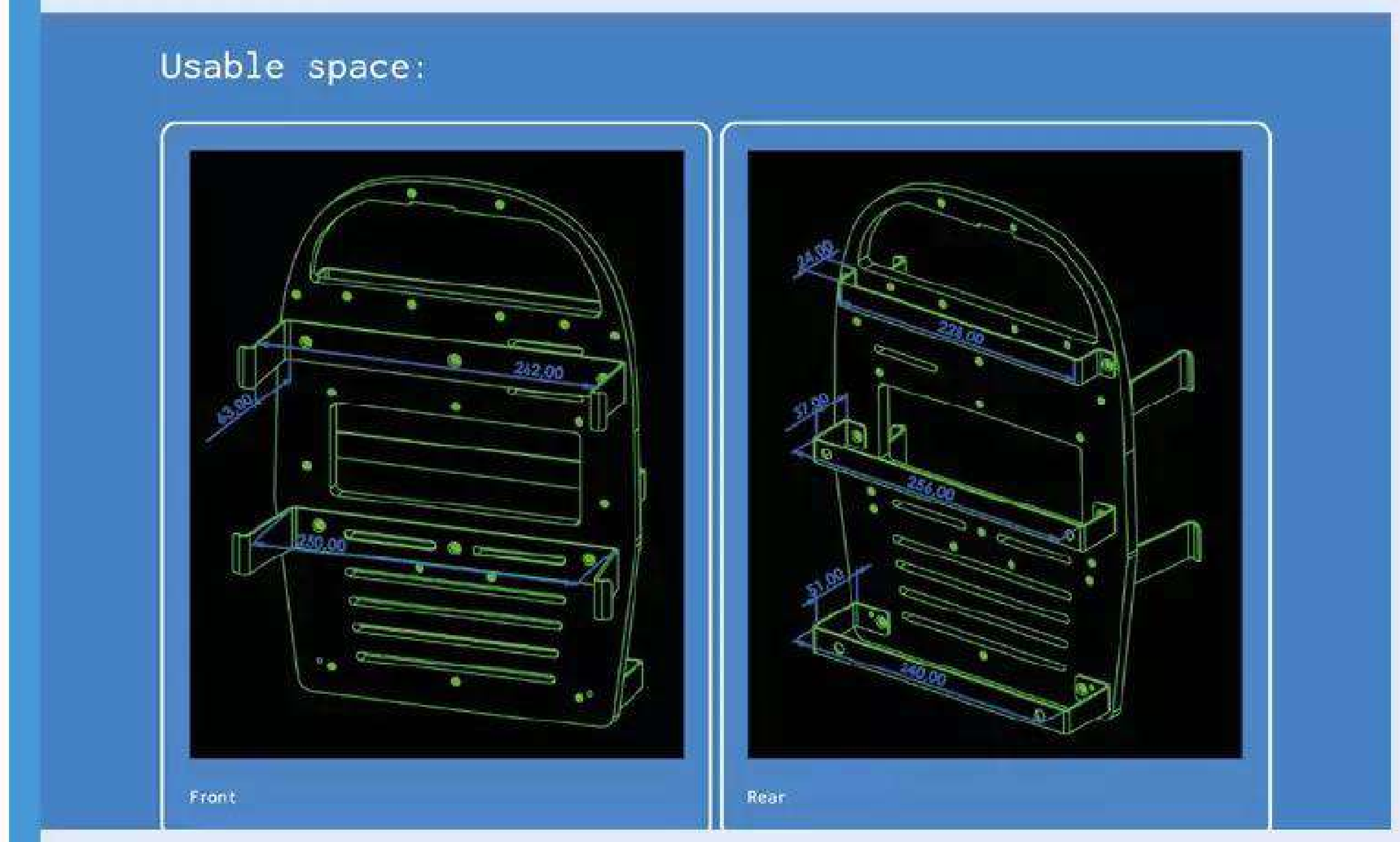
Reassuring curious but impressed project followers, Davide says all the components meet flight-safe guidelines for power output, so it could potentially be taken on board aircraft too. Davide is at pains to point out that everything he's done with the Backpack Cyberdeck is legal and poses no security issues. He has posted several assembly videos on YouTube that show the breadth of uses for his backpacks, including using one as a mobile radio station: **magpi.cc/bagbuildsYT**.

"The idea could be adapted for various other purposes, such as mobile video streaming, electronic repair setups, or even as a simple organiser," he says. "My advice to anyone interested in similar projects is to start small, and to pay attention to the interference that the various devices and cables could generate, compromising the functionality of some parts."

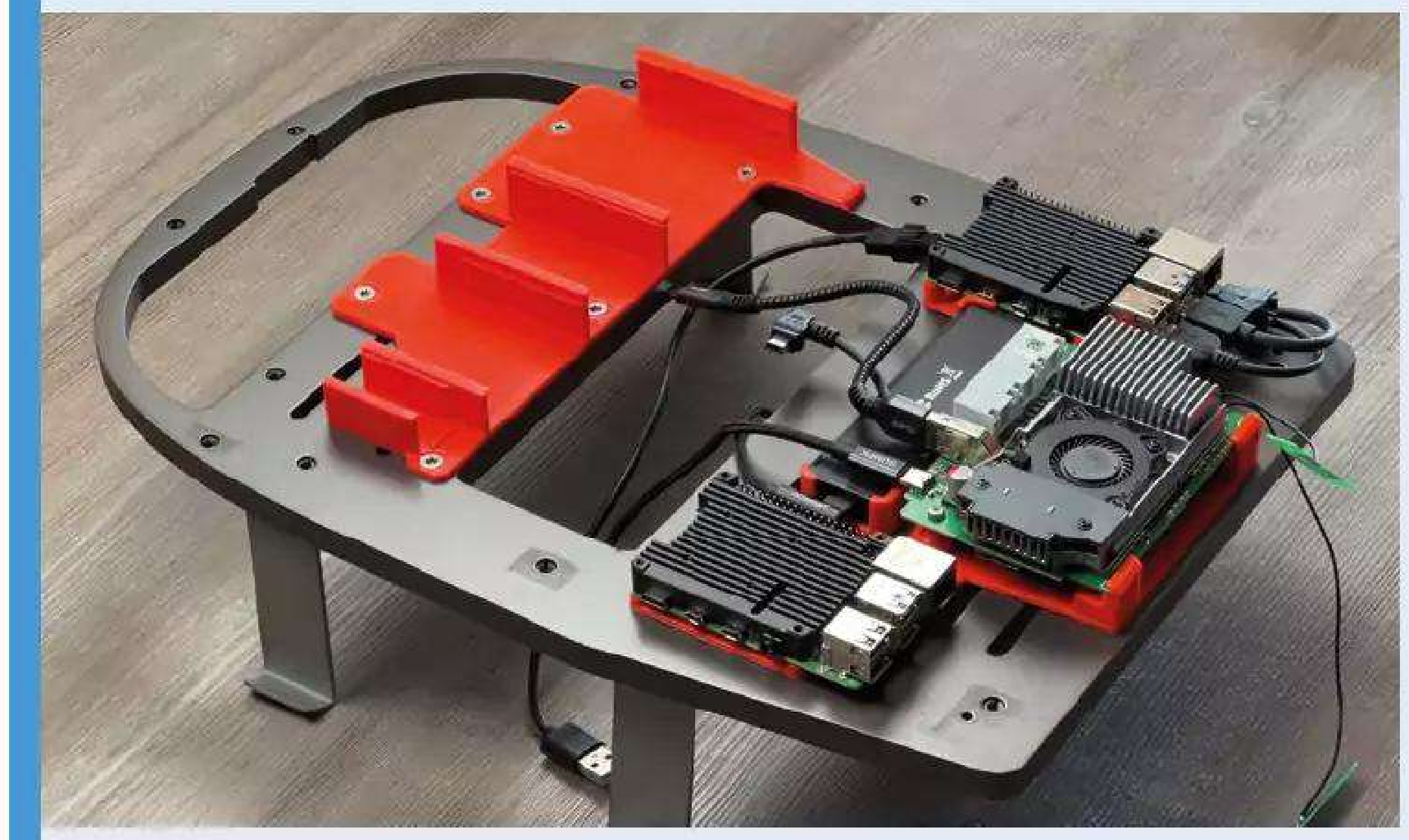
▼ Davide has now developed an online store for versions of his Backpack Cyberdeck



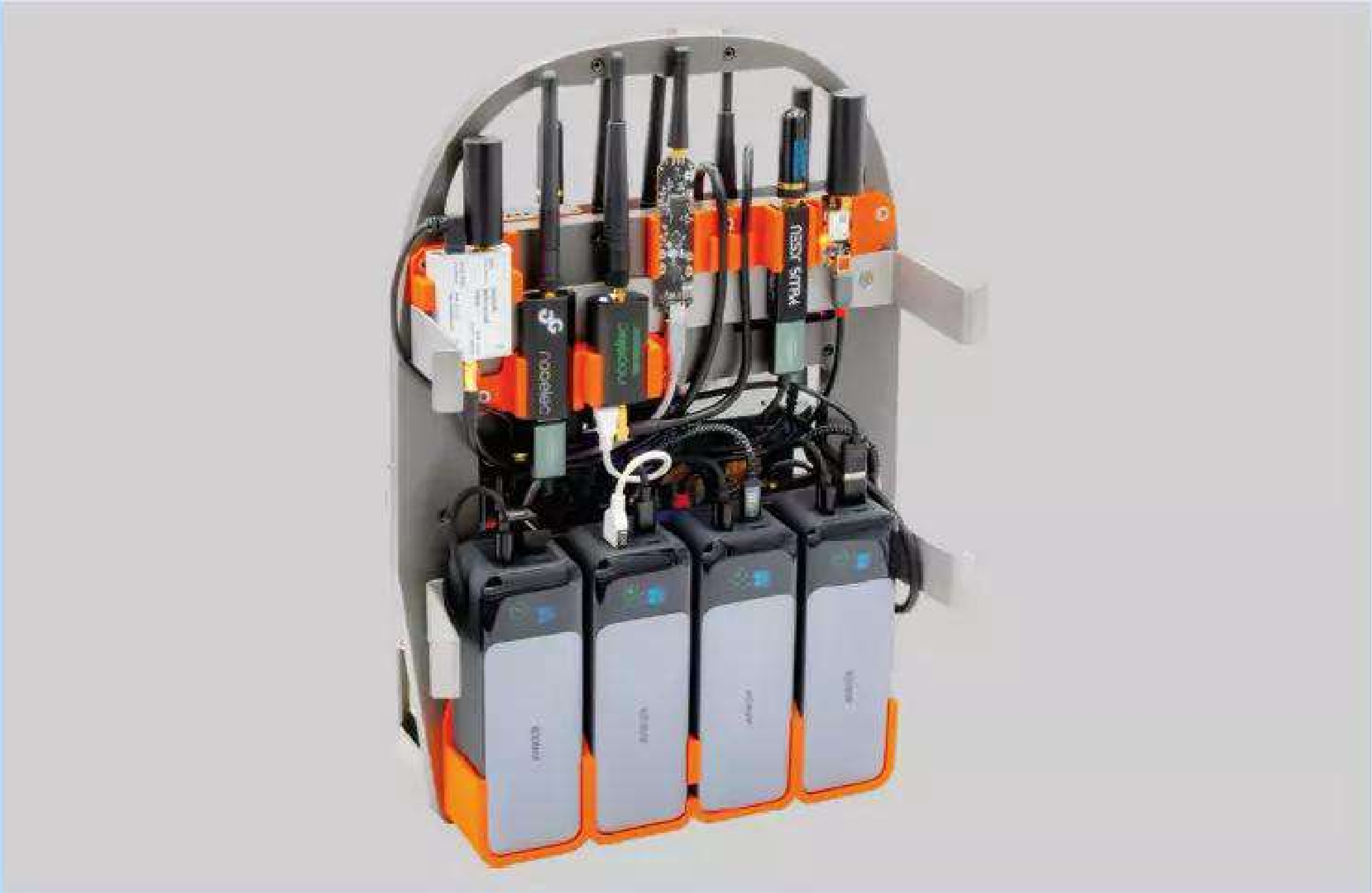
In the frame



01 Download and print (or design your own) backpack frame from Davide's online STL files at **magpi.cc/bagbuildsgit**. The frame will fit a standard backpack size and shape.



02 The cyberdeck setup is controlled by Raspberry Pi 4, with all-important heat dissipation components to keep the bundle of electronics cool and running optimally



03 The frame sits snugly inside the bag. Multiple USB devices and efficient and stable power packs support the array of sensing and communications hardware

Red Telephone

Dial M for Miles – Rob Miles to be exact. And when you do, don't be surprised if he answers you on an iconic 20th century landline phone, as **David Crookes** discovers

MAKER

Rob Miles

Rob Miles has been playing with hardware and software since almost before there was hardware and software.

magpi.cc/redtelephone

In the 1970s, people either headed for a phone box to make calls or signed a telephone contract with the General Post Office. If they did the latter, they'd invariably end up renting a shiny GPO 746 Rotary telephone available in a variety of colours. Callers would stick their fingers into the number holes, turn the dial anti-clockwise until it reached the finger stop, and let go. After the right number of digits, their call would connect.

Until button-press phones arrived this method was standard, and Rob Miles loved it. "The old phones were a huge part of my life when I was younger," he says. "Everybody had one. And everybody had the same one." He recently spotted one in a shop called Sellit & Soon in Beverly, England, and snapped it up. "I bought it on sight," he says. "Once I got it, I had to do something with it, so thought about putting a Raspberry Pi inside."

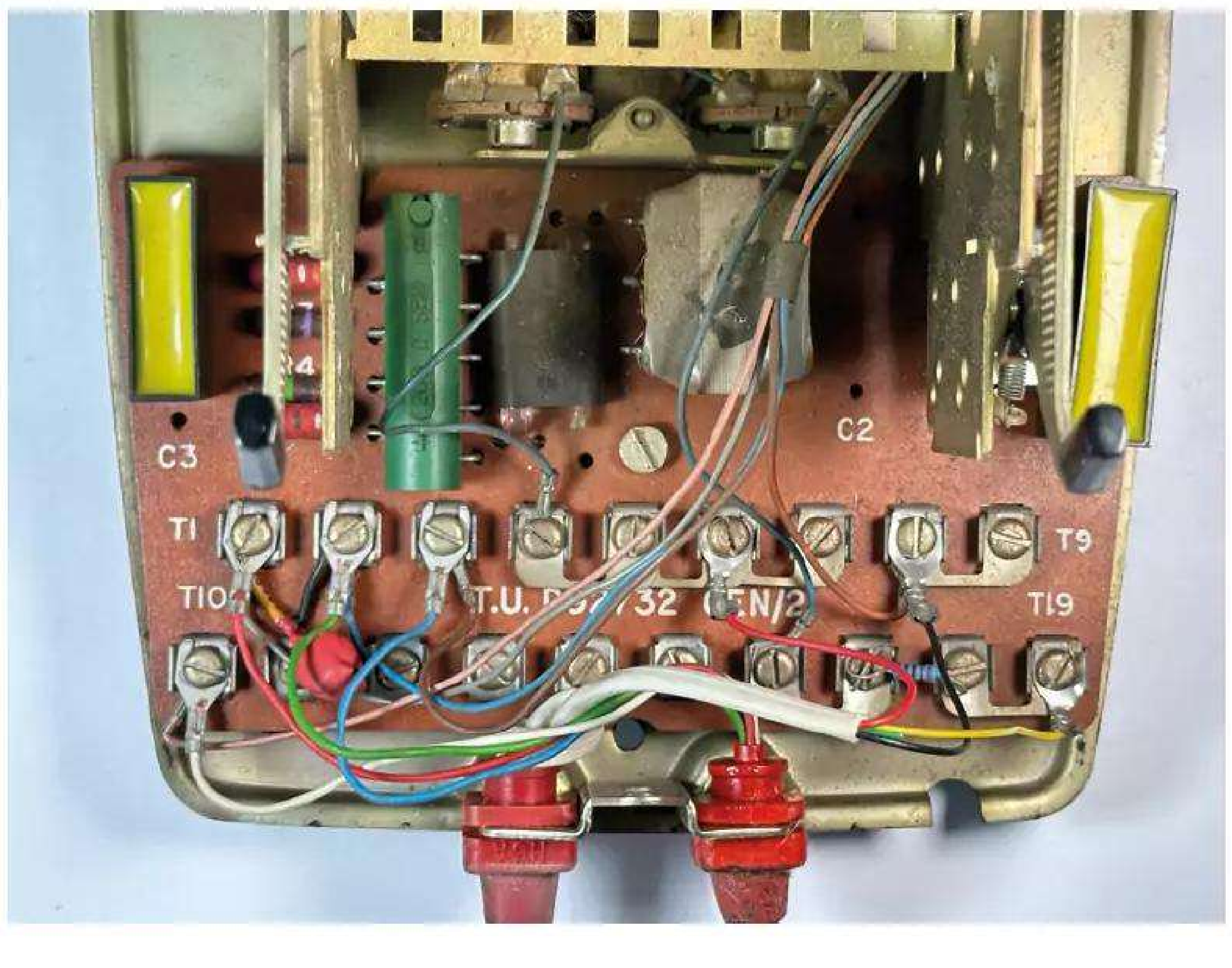
Holy moly!

Rob bought a red phone, not unlike the one used in the 1960s Batman television series. "I think the style is timeless," says Rob, who initially only aimed to make the bell ring. "I wanted it to sound exactly like the phones I remembered and I initially used a Raspberry Pi Pico to do it."

The idea was to get the bell to chime whenever the receiver was picked up and put down. Rob also wanted to be able to make the phone ring after dialling a number – and for the ringing to continue until the receiver was picked up. This entailed opening up the phone, stripping the internals and connecting Pico to a pair of MOSFET switches which sent high-voltage signals to the bell coils

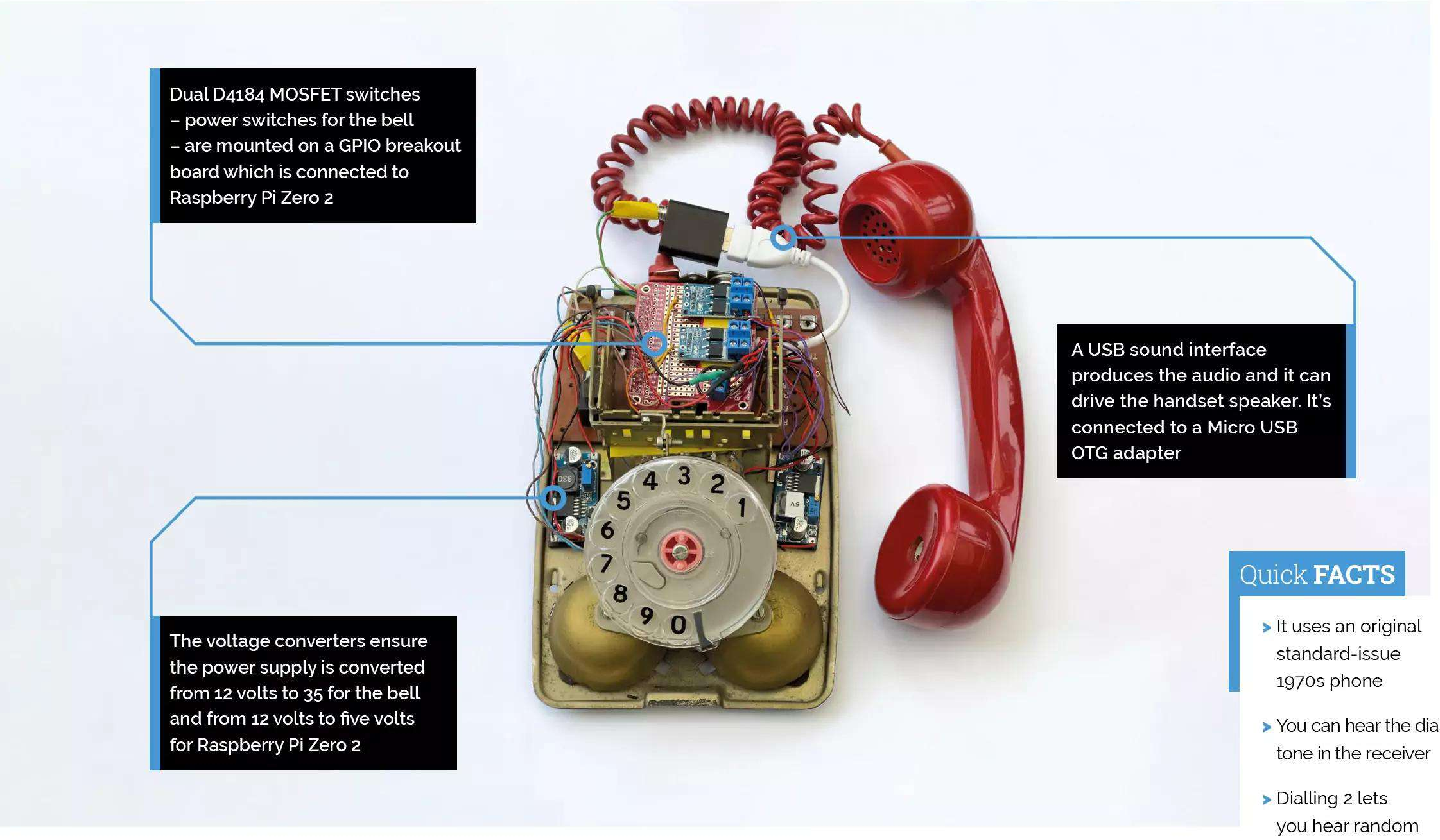
"Just making the bell ring posed all kinds of problems with power supplies but, because I wanted other folks to be able to build the thing with the minimum possible danger from high voltages, I worked out how to do it with a 35 volt supply rather than the 70 volts normally used."

Once Rob had the bell ringing, he decided to swap out Raspberry Pi Pico for Raspberry Pi



▶ These internal components are from the original phone, and include bell coils, resistor bulbs and an audio transformer – all essential for making and receiving calls

▲ Rob made sure the original casing remained intact so that, on the outside at least, it would be difficult to tell that the phone had been modified



Dual D4184 MOSFET switches – power switches for the bell – are mounted on a GPIO breakout board which is connected to Raspberry Pi Zero 2

A USB sound interface produces the audio and it can drive the handset speaker. It's connected to a Micro USB OTG adapter

The voltage converters ensure the power supply is converted from 12 volts to 35 for the bell and from 12 volts to five volts for Raspberry Pi Zero 2

> "I wanted it to sound exactly like the phones I remembered"

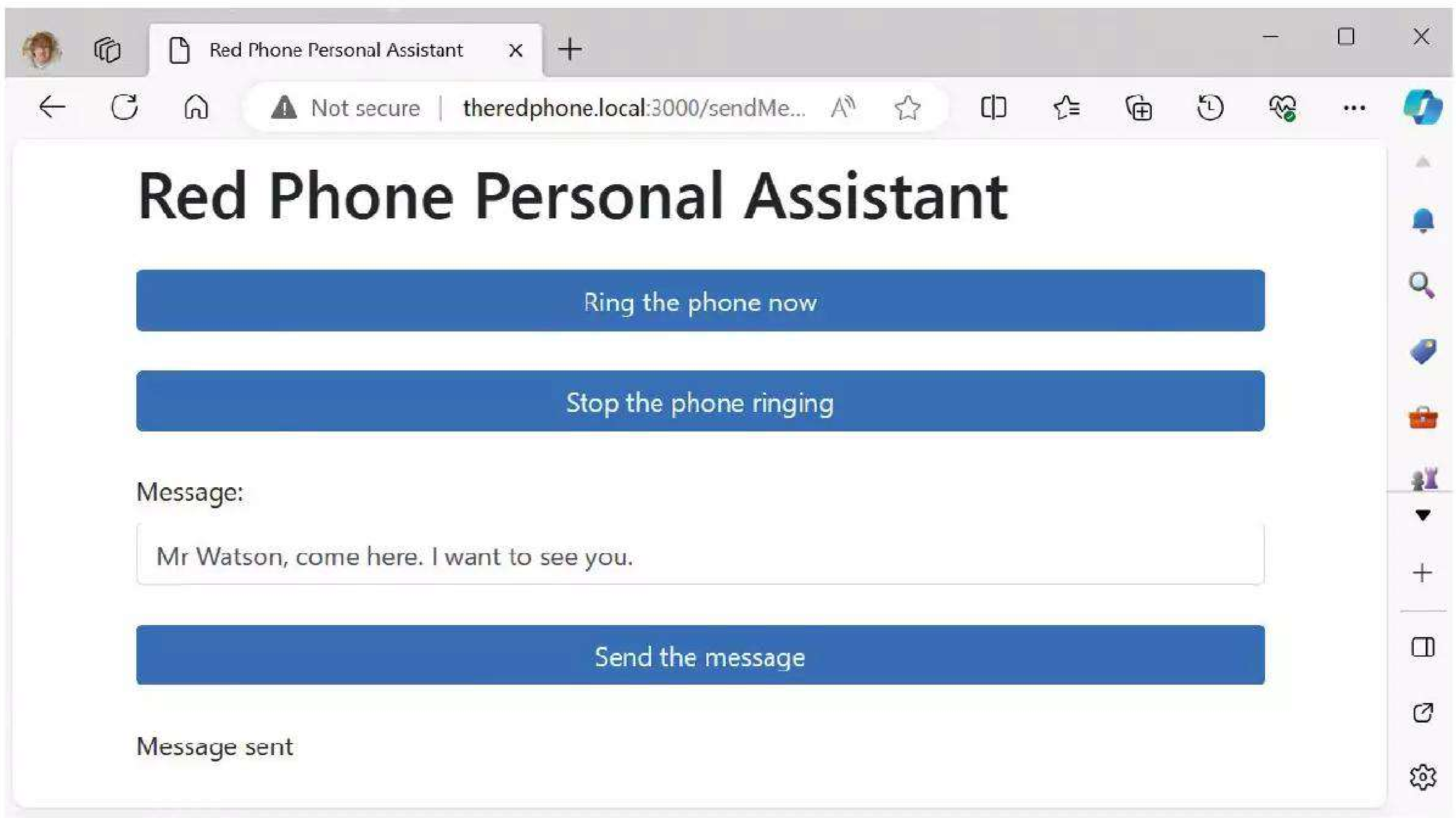
Zero 2 which, he says, gave the phone some serious processing power. Since he wanted a dial tone to play in the receiver, he had another idea. "I figured I had enough to do something that could read messages," he says. The phone was being turned into a robotic personal assistant.

Who's calling?

To allow the phone to receive messages and alerts, Rob used JavaScript running in the node environment to interact with the hardware. "Everything is controlled by a lump of JavaScript which runs under nodejs," he explains. "The program runs when Raspberry Pi boots and attaches event handlers to all the inputs – the handset switch and the two dial signals. It then waits for something to happen.

"At the same time it spins up an Express-powered website and announces itself on the local network so users can find it. Local actions – picking up the phone and dialling numbers – are handled directly by JavaScript. Remote actions such as ringing the phone speaking messages are assigned routes on the website to trigger the appropriate response." Rob can now type messages on a web page, have the phone ring and, when the receiver is picked up, have the messages read out. There's much potential here. "I can play back sound effects, which is how I get the dial tone to work, so we could have a lot of fun with this," he teases.

▼ The web interface controls the phone so you can make it ring and send messages to be heard from the receiver



Red Phone Personal Assistant

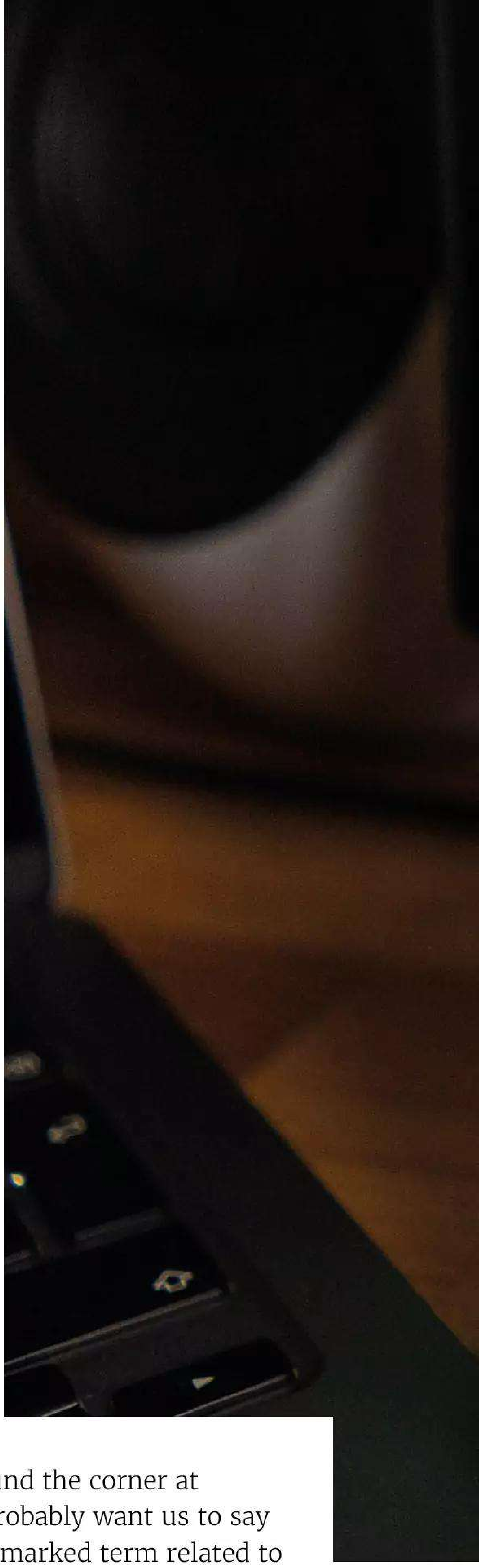
Message:

Mr Watson, come here. I want to see you.

Message sent

Quick FACTS

- It uses an original standard-issue 1970s phone
- You can hear the dial tone in the receiver
- Dialling 2 lets you hear random messages
- It's powered by Raspberry Pi running JavaScript
- Rob wants to add speech input



Automatic Speech Recognition Assistant

A truly mobile and portable smart assistant that uses AI tech on Raspberry Pi 4. **Rob Zwetsloot** asks its some questions

MAKER

Adam Frydrych

Adam helps the Meshnet community and is a maker passionate about all things tech, DIY, and Mazdas.

magpi.cc/autospeechrec

We've had voice assistants appear in *The MagPi* for a very long time – we even had a kit included with issue 57 you could use to build one yourself! Since then a *lot* has changed in the voice assistant, voice recognition, and AI chat space.

"[It] is my genuine belief that with moderation and caution, AI models are extremely useful," Adam Frydrych says to us. He's a Meshnet Evangelist (**magpi.cc/meshnet**) at Nord Security, and has created his own AI voice assistant using modern tech. "The ability to have a random question answered within seconds is amazing."

With this belief in mind, and inspired by Home Assistant's Year of the Voice initiative in 2023 (**magpi.cc/yearofvoice**), Adam went on a search for "alternatives to Rhasspy (**magpi.cc/rhasspy**) and other ready-made solutions for voice assistants," as he tells us.

Learning to talk

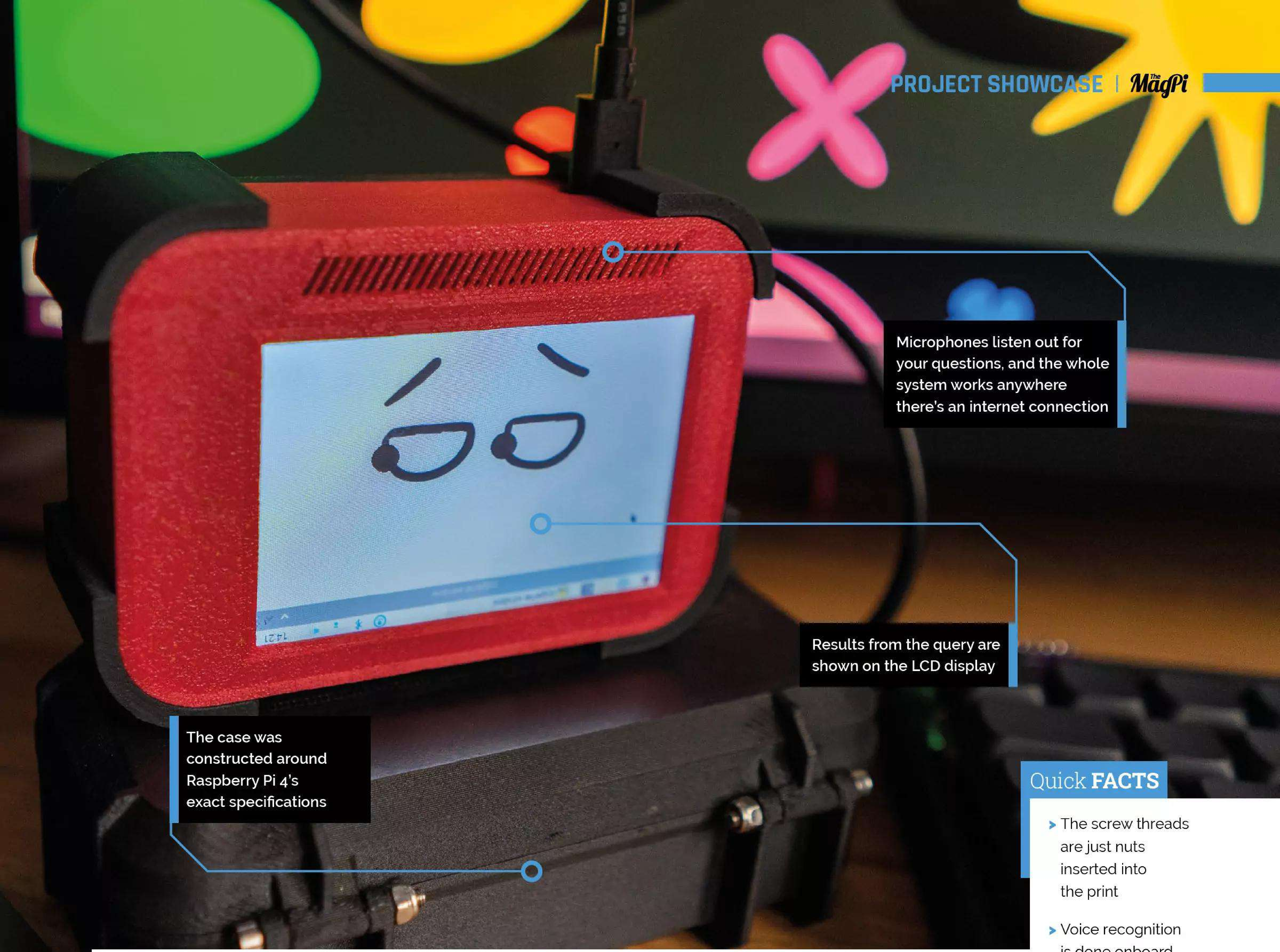
Making this on Raspberry Pi was to him a 'no-brainer'. "Raspberry Pi boards are reliable, familiar, and have unmatched software and hardware support," Adam says. "Raspberry Pi boards 'Just Work™'."

The important folks around the corner at Raspberry Pi Towers will probably want us to say that's not an actually trademarked term related to Raspberry Pi but we at The MagPi appreciate the enthusiasm and whole heartedly agree with how easy it is to make something on one.

"The construction of the project had a few iterations," Adam explains. "Luckily, I was able to find 3D models of Raspberry Pi that allowed me to pack components tightly in a slim 3D-printed shell. I also wanted to make sure the project would be affordable, so that's why I opted for a second-hand PlayStation Eye camera that sports a four-microphone array for a whooping cost of €5."

The PlayStation Eye was a PS3 accessory – not to be confused with the EyeToy on PS2 – and can capture video at a very high framerate (187 hertz in some use cases) as well as having a fairly advanced microphone array for a console webcam. It also helps that it's very, very obsolete, resulting in its low price.

"The project employs Meshnet in order to access the language model that's hosted on my home lab server or my desktop PC," Adam says. It's a feature of NordVPN that allows you to set up encrypted connections between your devices. "This way, I can run bigger models and have lower inference times,



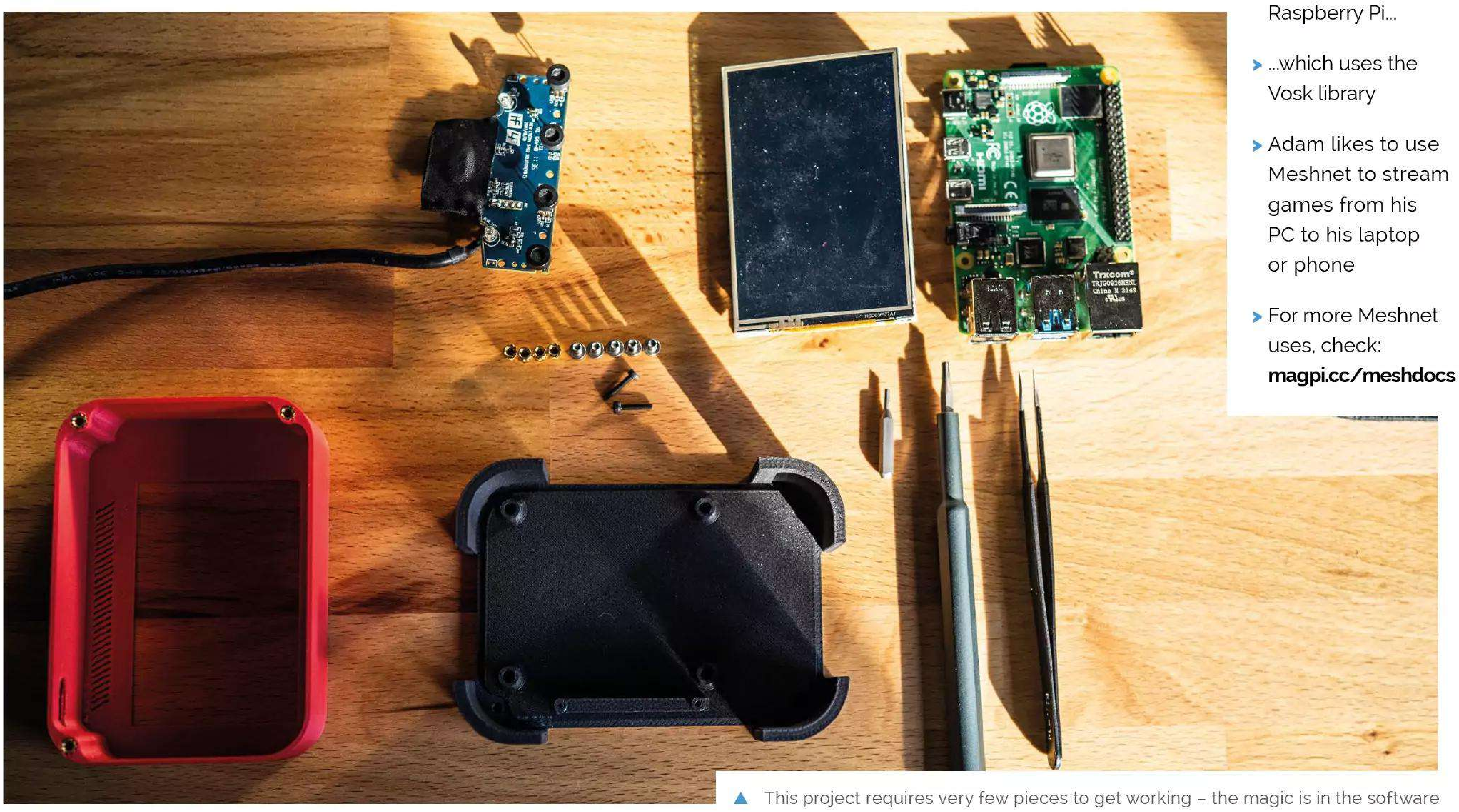
Microphones listen out for your questions, and the whole system works anywhere there's an internet connection

Results from the query are shown on the LCD display

The case was constructed around Raspberry Pi 4's exact specifications

Quick FACTS

- The screw threads are just nuts inserted into the print
- Voice recognition is done onboard Raspberry Pi...
- ...which uses the Vosk library
- Adam likes to use Meshnet to stream games from his PC to his laptop or phone
- For more Meshnet uses, check: **magpi.cc/meshdocs**



▲ This project requires very few pieces to get working – the magic is in the software

◀ The final product looks very smart we think – we could see several of these dotted around a house

▼ Soldering a USB device to a Raspberry Pi is a bit of a last resort for some projects – it can be dangerous and you risk damaging your Raspberry Pi. It works well though, when done right.

> “The project employs Meshnet in order to access the language model that's hosted on my home lab server or my desktop PC”

◀ There's no battery inside this version, so you'll need to provide external power from a plug or a mobile battery especially when using my GPU. Essentially, every device in question runs Meshnet, and depending on what device is currently running the AI model, I can easily swap between them, even when I'm away from home."

Proofreading

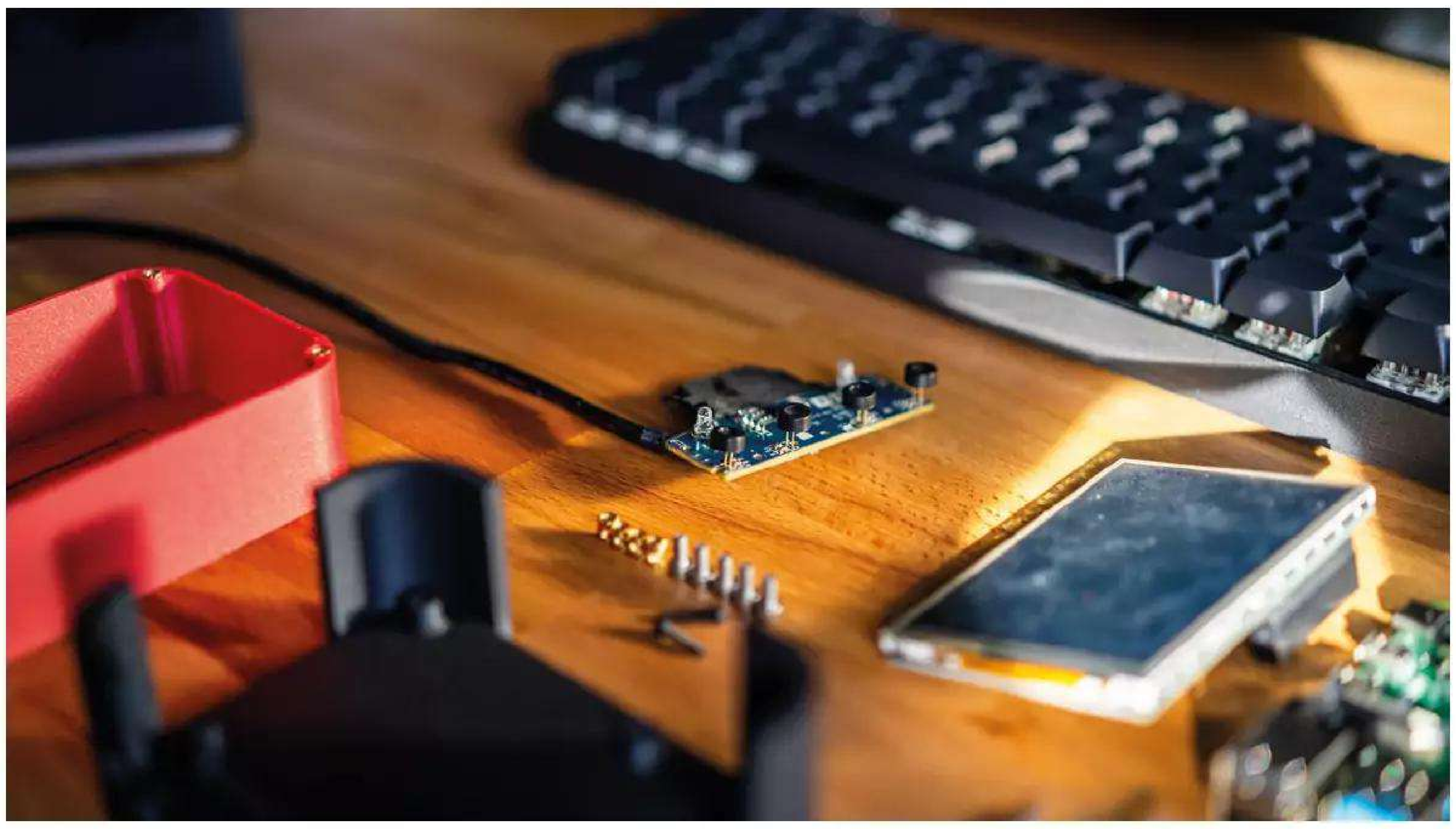
According to Adam, as it stands it works 'as expected'. The voice samples are correctly relayed and analysed, and a response is displayed screen.

"There is, however, room for improvement," he admits. "The AI model I have used in the past, Luna, is quite slow compared to a language model like Phi-2. The response times go down dramatically with Phi-2. Another thing that could be improved is waiting for the request to be responded to before returning to listening for another prompt."

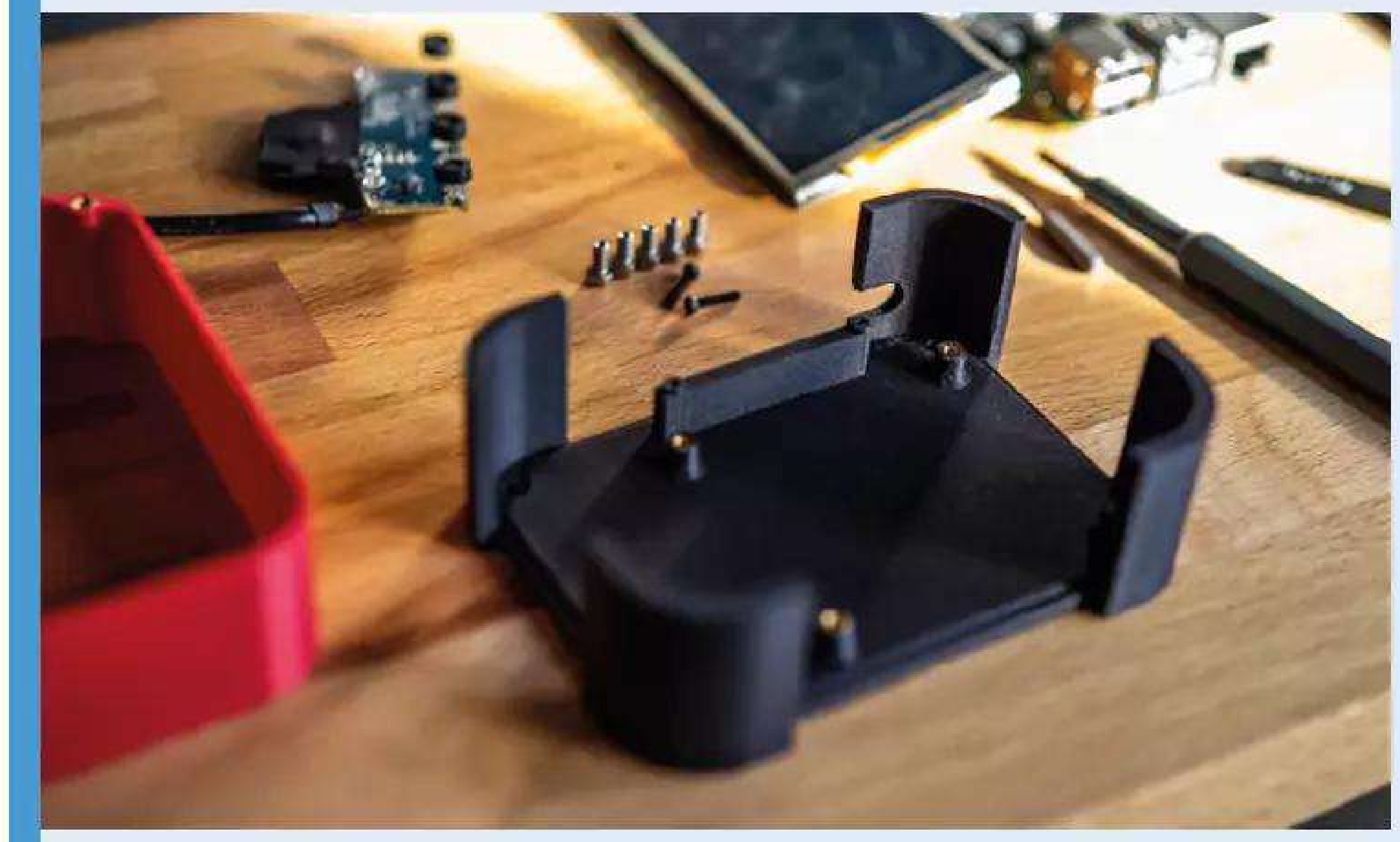
Responses to it on Hackaday have been largely positive, which is always a good sign, however he still has other upgrade plans such as moving to a Raspberry Pi Zero 2 W, getting an even better microphone, and tweak the timing on some of the functionality.

"The satellite starts listening for another prompt before the request from the language model returns," Adam tells us regarding timing. Seems a bit eager to us.

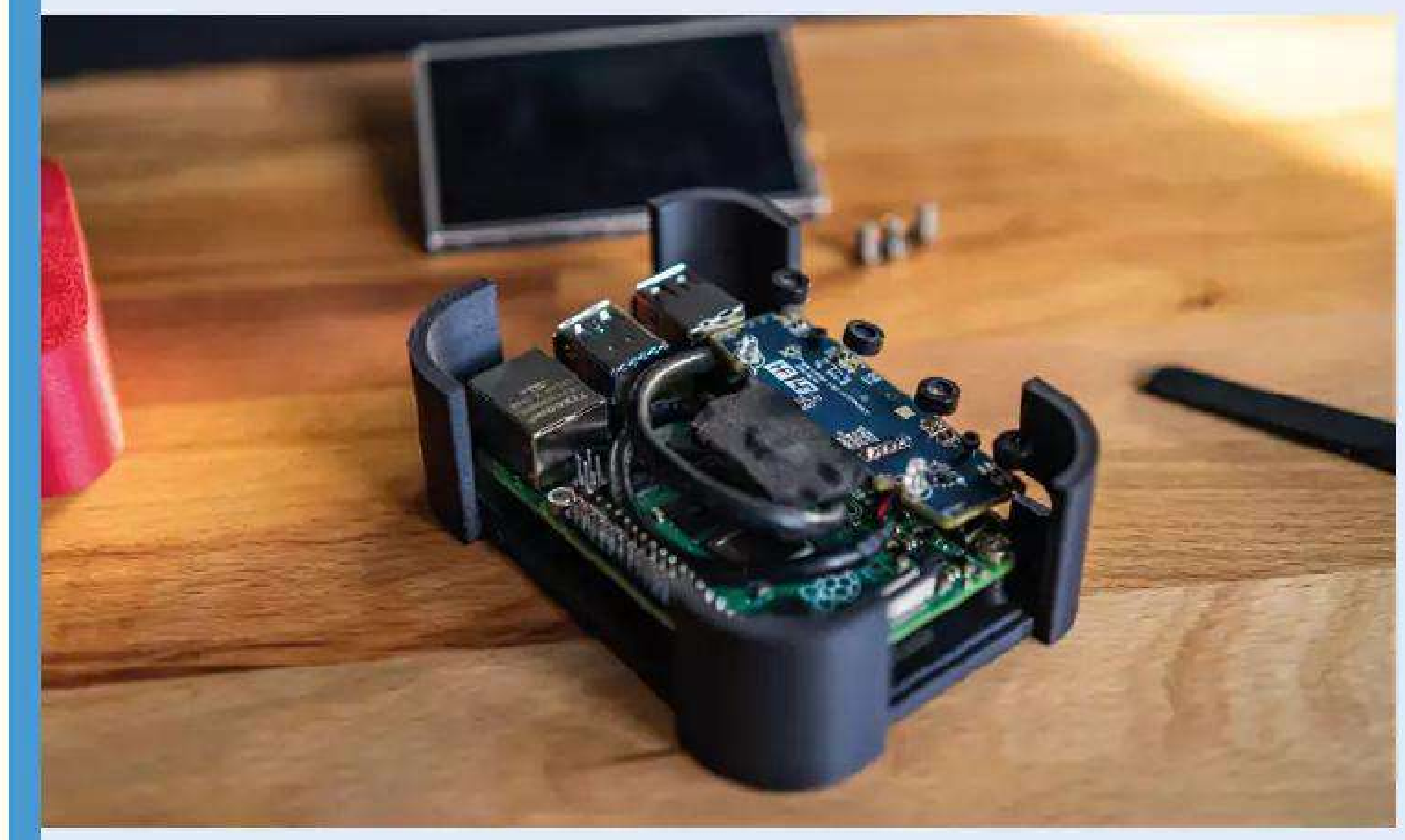
▼ The board for the PlayStation Eye is very small and good enough for the job here



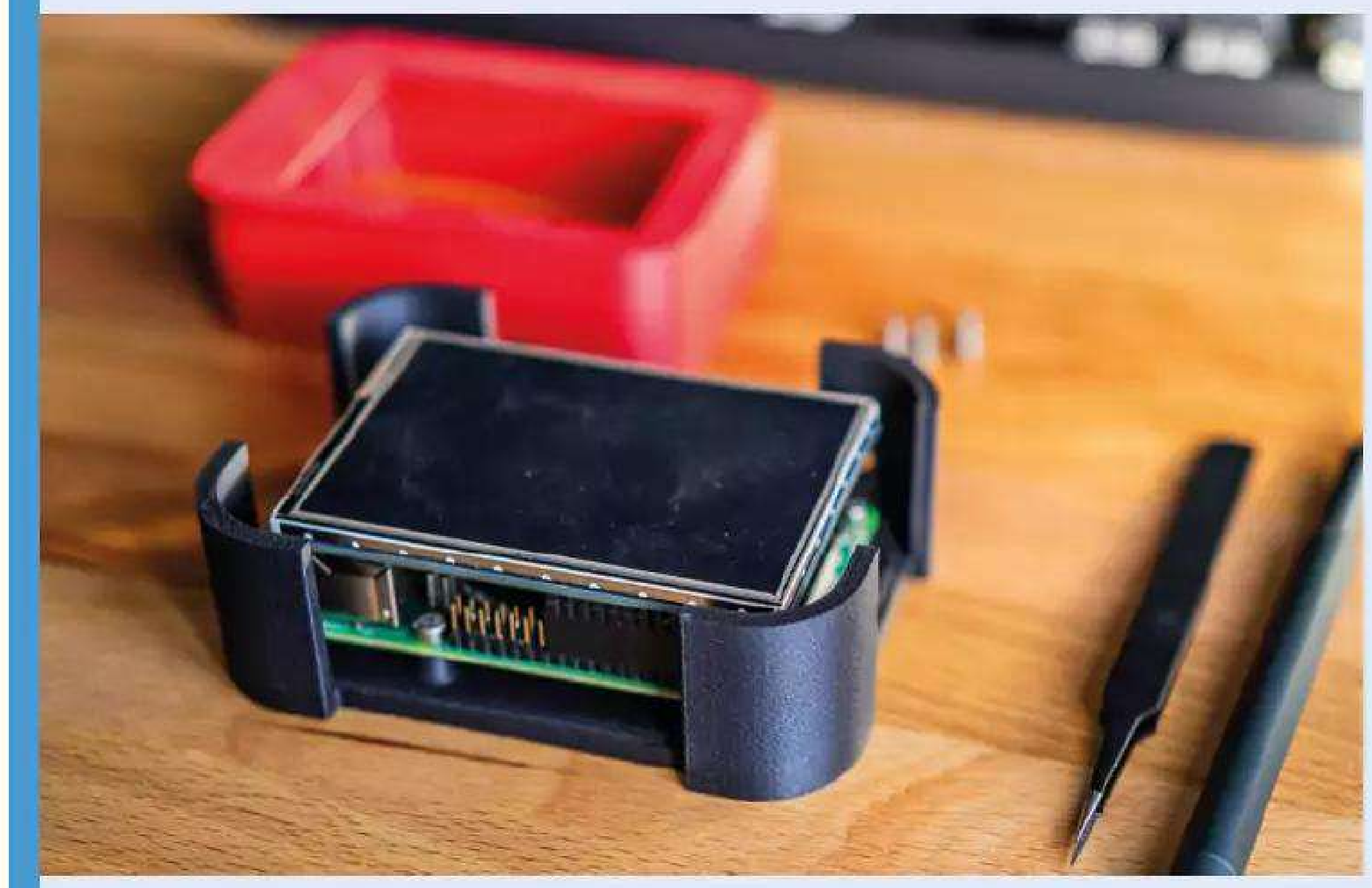
Build a voice assistant



01 This case is built around a 3D model of Raspberry Pi and the screen. The tolerances on a print like this can be tight, and you may need to experiment a bit to get it just right. Also, threaded screw holes make it a little more robust.



02 The PlayStation Eye's main board, which includes the microphones, is soldered to the USB port on the underside to save space. The camera is also taped over because it won't be needed, don't worry about it.



03 Finally the screen is added, It's an older TFT screen that sits on the first 26 pins of the GPIO and shows the assistant's face and its answers to your questions. Putting the top section on completes this (fairly simple!) build.

Mini Observatory

Finally having the time to focus on his hobbies led one maker to design and build a self-contained star-gazing setup, learns **Rosie Hattersley**

MAKER

Matt Hough

Matt Hough is an ardent Raspberry Pi aficionado and astronomy fan using his recent retirement to focus on ever more complex makes

shortbus.blog

Warning!
Stay indoors

This is a tabletop observatory for indoor use and the dome is not weatherproof. (Matt plans one that is). This version should not be used outside as the electrical components will be damaged

magpi.cc/electricalsafety

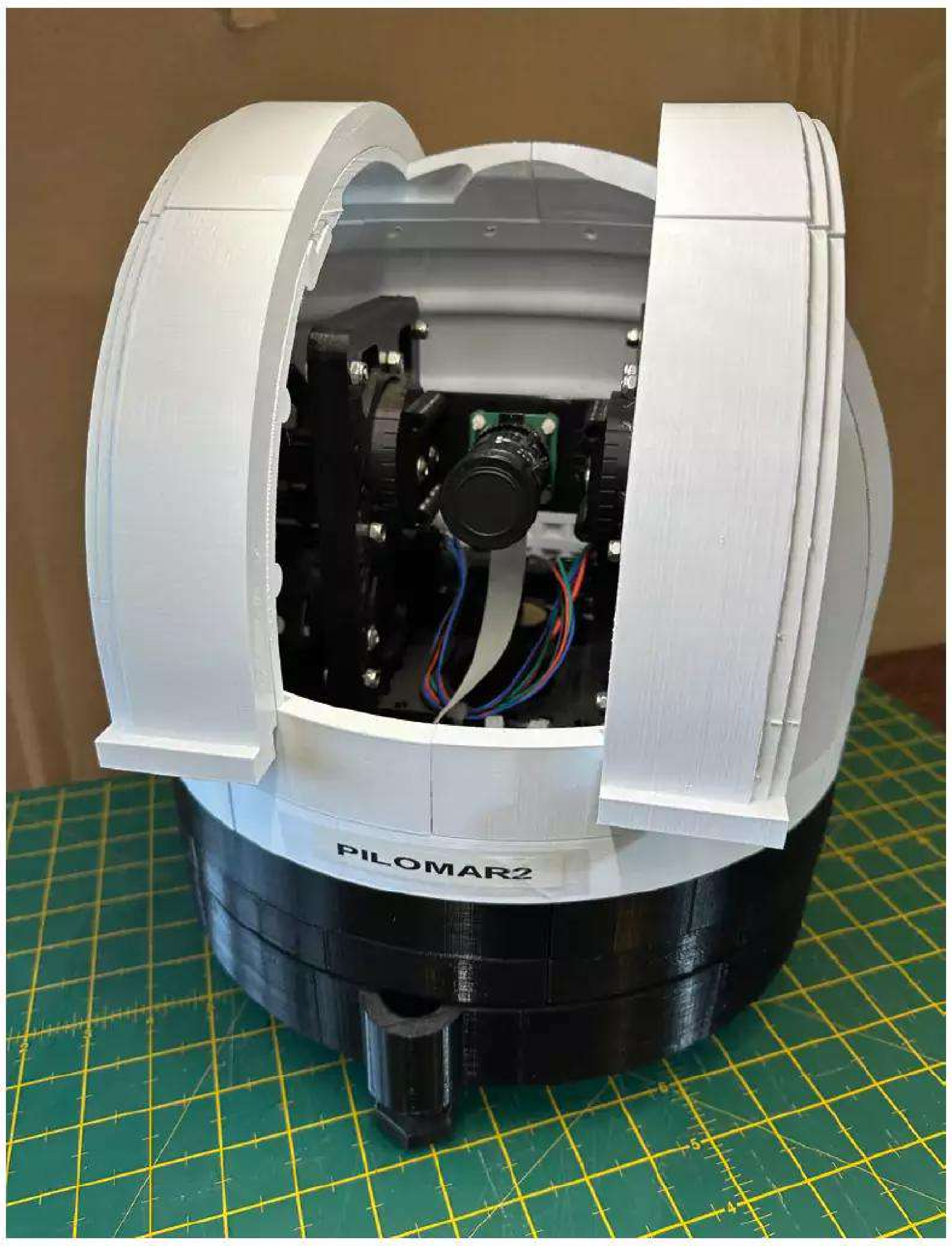
Matt Hough's retirement from the corporate IT world freed him up to return to hobbies that fascinated him many years ago. He studied computing in the 1980s, but had not tinkered much since, other than with software.

While astronomy had always been a big draw, Matt rarely had time to pursue it while working. When Raspberry Pi was launched in 2012, Matt was immediately drawn to it because "it was incredibly easy to make and experiment with". After familiarising himself with Python, Matt found himself buying several more Raspberry Pi computers Matt found himself appreciating their diminutive size too: his new hobby was portable! "I could pack a small kit and take it with me when travelling – so whenever I had downtime, I could tinker and try things". Several of the projects he created have been astronomy-themed, but creating a mechanised mini observatory and telescope based around Raspberry Pi 4 offered far more of a challenge – one that took two years of development and would see him create a dozen versions of his telescope before he felt it was ready to be shown off to the world. Once he did, Matt was "slightly stunned how much interest it generated". We think it's pretty impressive.

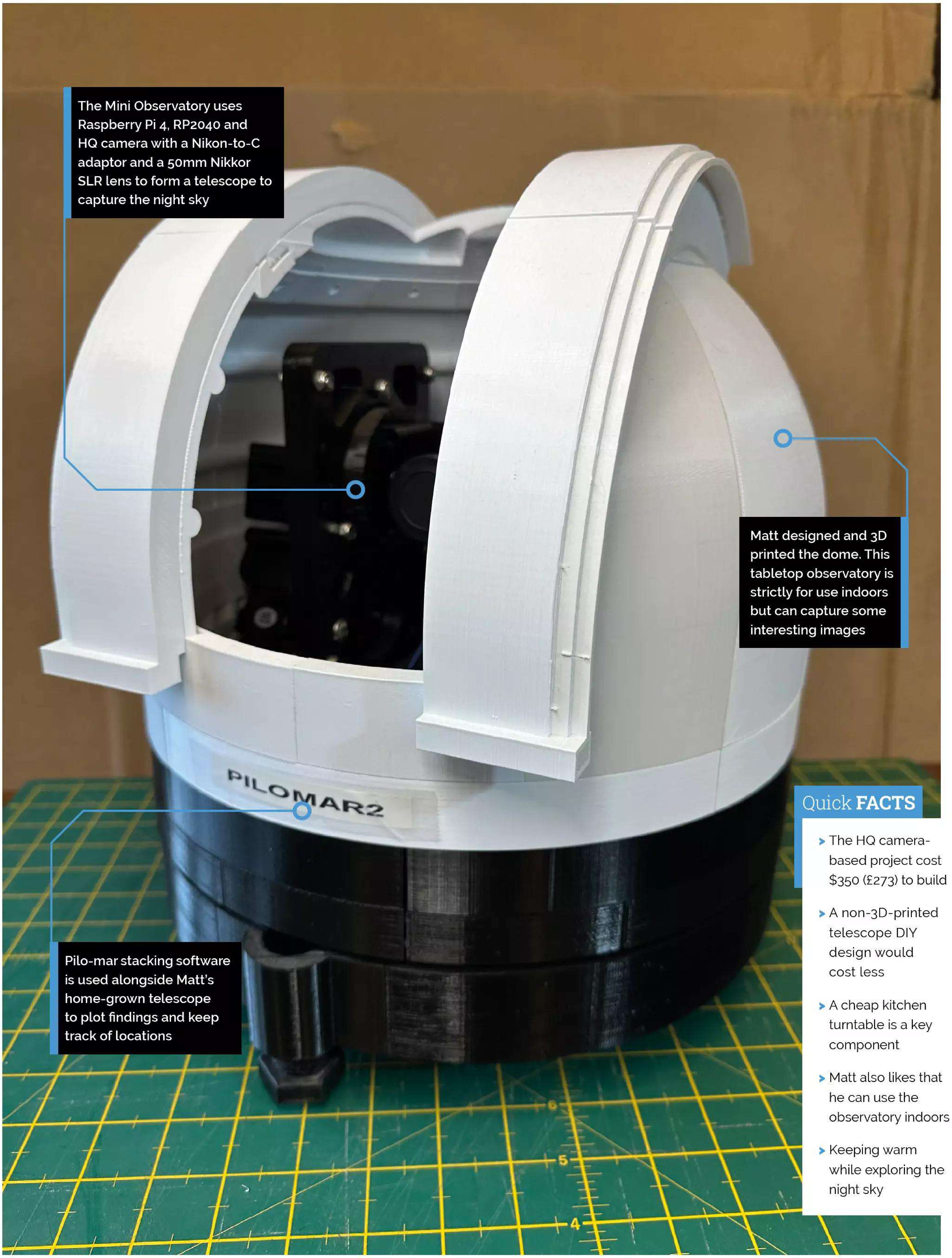
A familiar tale

Matt began with a series of traditional telescopes, but found them "a little frustrating to use in practice". It was hard to locate objects in the sky and most of what he could see was "often just a grey smudge". Digital telescopes were not that common and were expensive, but when Raspberry Pi launched the HQ camera sensor Matt wondered whether he could build a really simple digital telescope with the Raspberry Pi at the heart of it.

He had already built an Aurora clock that lets you know if there's a chance of seeing the Northern Lights, and a similar device using Raspberry Pi to track the International Space Station. With ten years of Raspberry Pi familiarity to draw upon, Matt was confident he could handle both high- and low-level functions and decided it would be the ideal basis for his own telescope and observatory design. "I knew there were Python packages available and I hoped to find pre-existing solutions to most of my needs". The initial plans were for a Raspberry Pi Pico project but Matt soon realised he needed more power, switching to "Raspberry Pi 4B with RP2040 help".



▲ Matt viewed designing a telescope as a great educational project



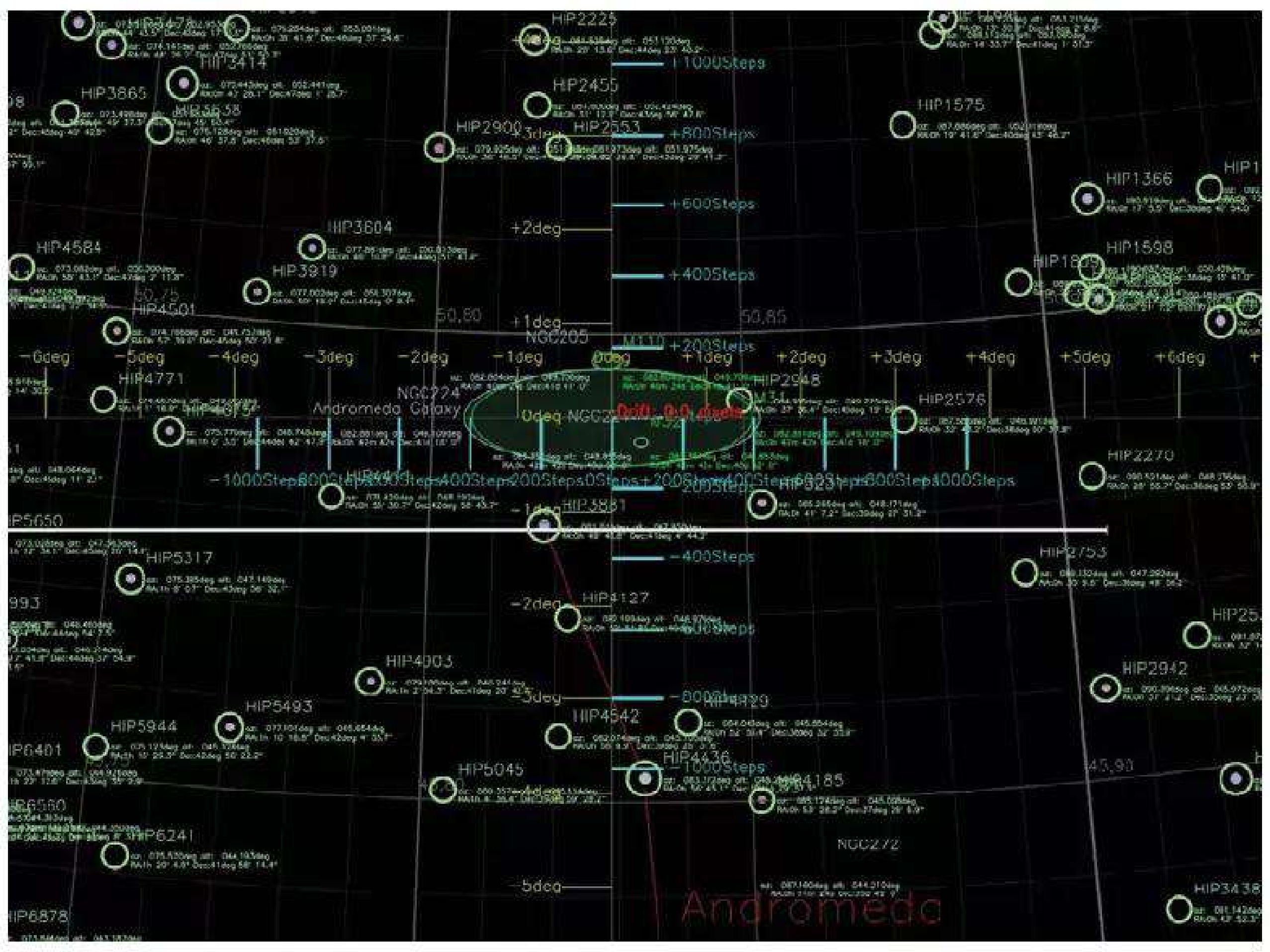
The Mini Observatory uses Raspberry Pi 4, RP2040 and HQ camera with a Nikon-to-C adaptor and a 50mm Nikkor SLR lens to form a telescope to capture the night sky

Matt designed and 3D printed the dome. This tabletop observatory is strictly for use indoors but can capture some interesting images

Pilo-mar stacking software is used alongside Matt's home-grown telescope to plot findings and keep track of locations

Quick **FACTS**

- The HQ camera-based project cost $350 (£273) to build
- A non-3D-printed telescope DIY design would cost less
- A cheap kitchen turntable is a key component
- Matt also likes that he can use the observatory indoors
- Keeping warm while exploring the night sky



◀ A processed photo of the Andromeda galaxy using images gathered by the telescope using a 50mm lens

▼ The software attempts to recognise objects and helps 'tune' the system

> “ He gave the motors a little RP2040 microprocessor brain ”

Matt was less confident of his 3D design skills: he needed to create more parts than he'd ever done previously and regarded this aspect as a distraction at the time, but says it's a new skill that has come in useful elsewhere since.

Seeing the lights

The Mini Observatory design emerged from successive experiments using Raspberry Pi Pico. Having started tinkering, Matt gradually developed routines to solve specific problems, learning how to control stepper motors and the technical aspects of the HQ camera. He needed precision here and went to a specialist company called Stepperonline (**omc-stepperonline.com**) to source them. Matt bought other parts from well-known Raspberry resellers such as The Pi Hut and Pimoroni, with more generic nuts and bolts from general hardware stores. He cautions over scrimping on potentially dangerous items such as power supplies: “always buy them from a trusted source”.

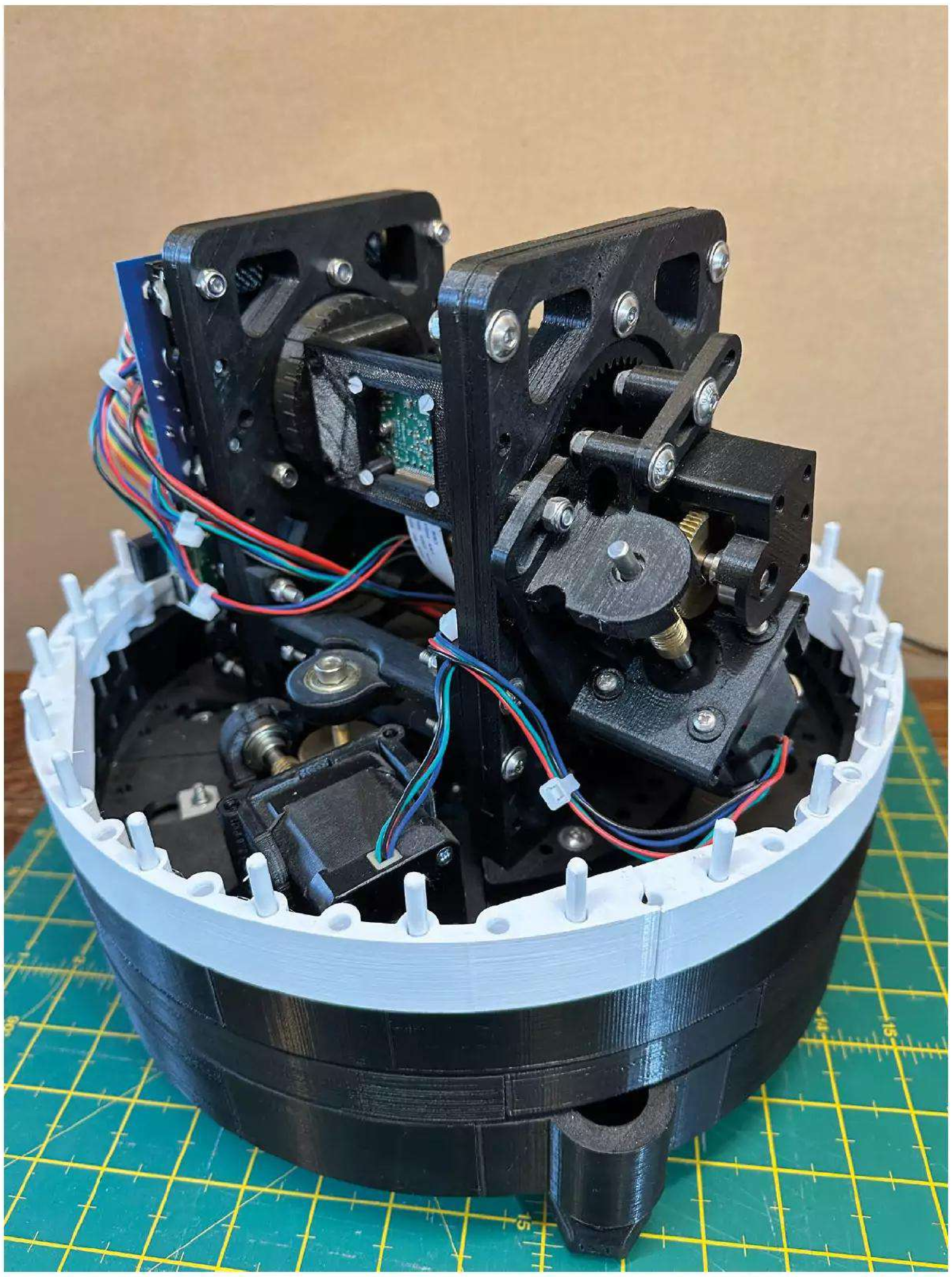
The project provided a great learning experience, with the observatory, gears and mechanism all home-grown. The ‘semi-intelligent’ motor controller for the telescope is probably the most novel element. Matt needed a way for the telescope to move while Raspberry Pi was busy taking photographs, so gave the motors a little RP2040 microprocessor brain. “They were released at the perfect time.”

Matt was able to make use of Python packages such as Skyfield, OpenCV, PiDNG and Astroalign and says it was a good choice for his Mini Observatory project. He is also really keen to process the photographs in real time onboard the telescope. “I haven't solved that yet, so I still need to do some offline processing afterwards. Realtime processing must be possible, I just have to research more.”

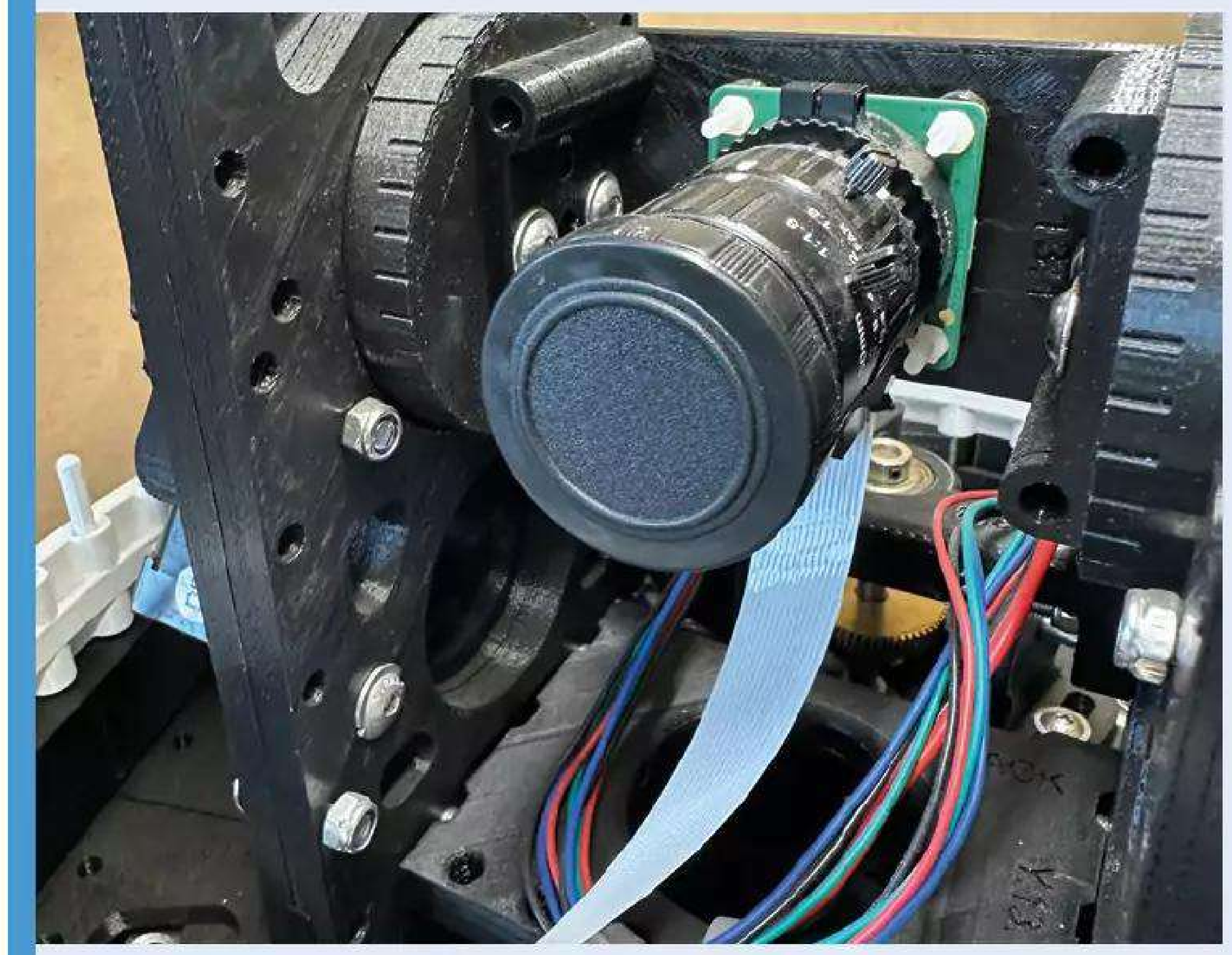
Since Matt first unveiled the project (**magpi.cc/miniobservatory**), several other makers have created versions, providing invaluable feedback and prompting him to tweak a few elements such as removing the infrared cutoff filters from his cameras which will make more objects visible.

Matt is also plotting a second version of his mini observatory, and is excited about the possibilities of Raspberry Pi with improved imagery and support for a second camera: one camera to do the tracking another to photograph the heavens. In fact "Raspberry Pi 5 may trigger quite a rewrite!"

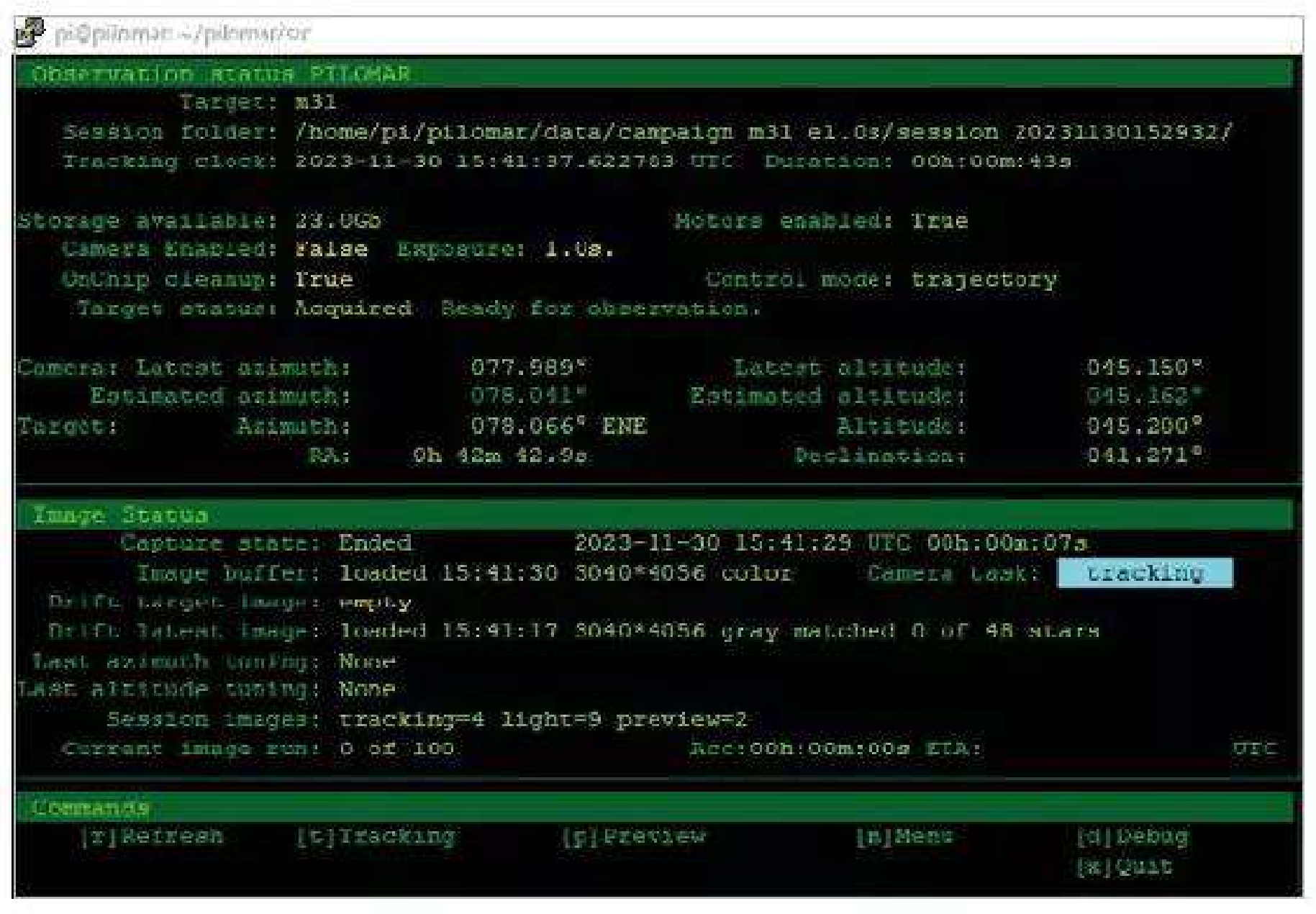
▼ The gearing allows the telescope to move vertically and horizontally in 0.004-degree steps.



Be observant

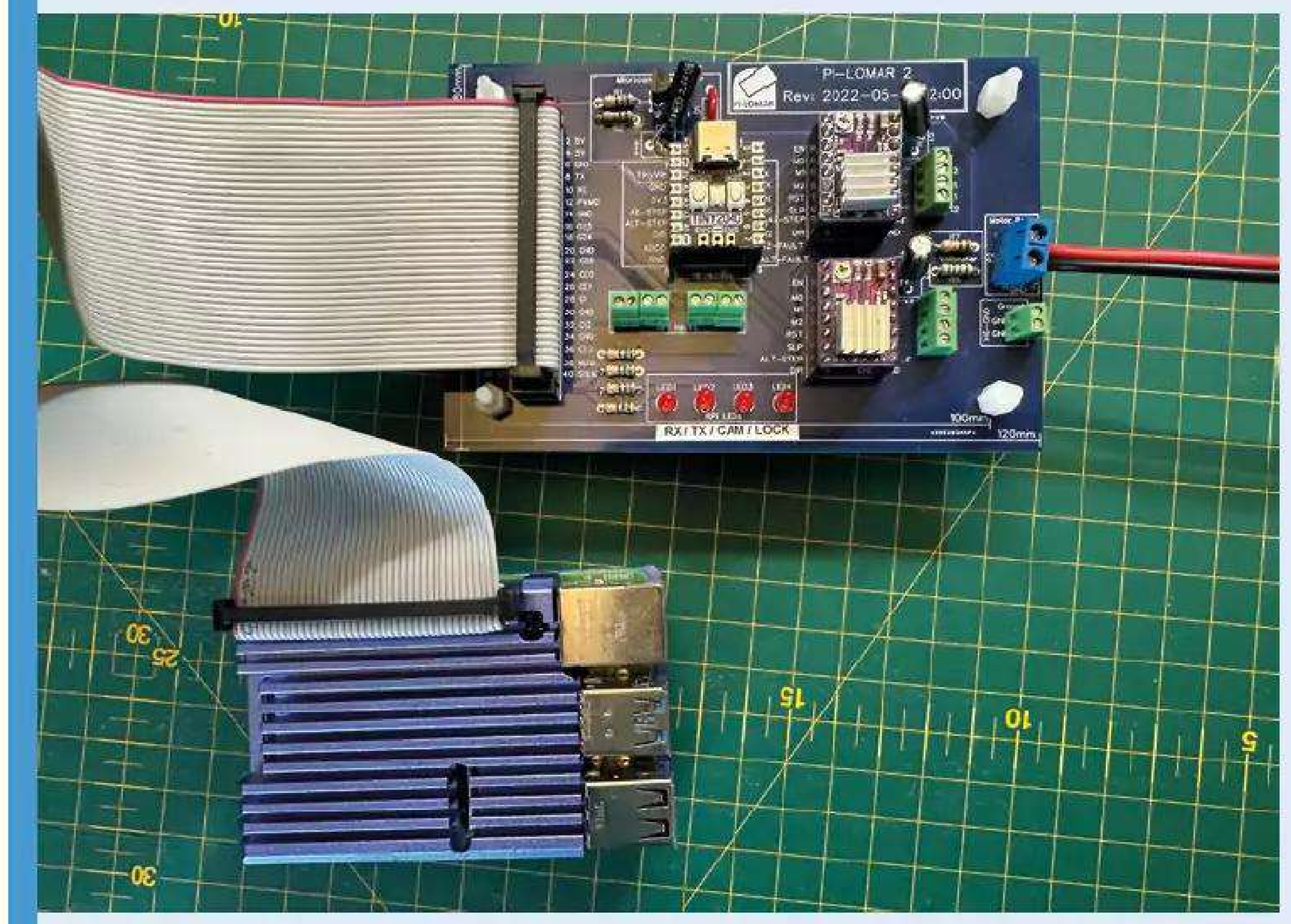


01 This project proves you can use Raspberry Pi camera components to make a working telescope. Full build instructions and STL files are on Matt's GitHub (**magpi.cc/pilomar**), but there is plenty of scope to design your own dome.



```
pi@pilomar: ~/pilomar/cr
Observation status PILOMAR
           Target: m31
   Session folder: /home/pi/pilomar/data/campaign m31 e1.0s/session 20231130152932/
   Tracking clock: 2023-11-30 15:41:37.622763 UTC  Duration: 00h:00m:43s

Storage available: 23.0Gb                    Motors enabled: True
   Camera Enabled: False  Exposure: 1.0s.
   OnChip cleanup: True                        Control mode: trajectory
    Target status: Acquired  Ready for observation.

Camera: Latest azimuth:        077.989°        Latest altitude:        045.150°
     Estimated azimuth:        078.041°     Estimated altitude:        045.162°
Target:         Azimuth:        078.066° ENE            Altitude:        045.200°
                    RA:     0h 42m 42.9s             Declination:        041.271°

Image Status
       Capture state: Ended            2023-11-30 15:41:29 UTC 00h:00m:07s
        Image buffer: loaded 15:41:30 3040*4056 color     Camera task:  tracking
  Drift target image: empty
  Drift latest image: loaded 15:41:17 3040*4056 gray matched 0 of 48 stars
 Last azimuth tuning: None
Last altitude tuning: None
      Session images: tracking=4 light=9 preview=2
   Current image run: 0 of 100               Acc:00h:00m:00s ETA:              UTC

Commands
    [r]Refresh     [t]Tracking      [p]Preview        [m]Menu       [d]Debug
                                                                    [x]Quit
```

02 Matt recommends Raspberry Pi 4 and Buster OS with at least 2GB RAM and a 32GB SD card "because Pilo-mar creates a lot of data" while tracking the skies



03 Pimoroni Tiny2040 and TinyPython control the azimuth stepper motor, while Raspberry Pi 4B oversees the camera tower that forms the telescope

Peltier cooler

We all know Raspberry Pi is cool but Ivan Kuleshov has created a striking way to keep the tiny computer even cooler, as **David Crookes** explains

MAKER

Ivan Kuleshov

Ivan is a systems engineer and hardware hacker. He runs Uptime Industries which has been working on Compute Blade, a 1U carrier board for the Raspberry Pi CM4.

uplab.pro

Raspberry Pi 5 boards can become hot and, if things become a little too roasty toasty – that is, if the temperature reaches 80°C under a sustained heavy load – the board will begin to throttle the processor. Since this limits the CPU speeds, the device won't run at its best, so it's always a good idea to try and avoid such a situation if you can.

The easiest and cheapest way to do this is to buy the official Raspberry Pi Active Cooler which costs less than a fiver. This single-piece anodised aluminium heatsink with an integrated blower keeps the idle temperature low, while springing into action when Raspberry Pi reaches a temperature of 60°C to help the device cope with high demands.

But if that seems like too simple a solution, you could follow the example of Ivan Kuleshov and create a cooler of your own. In this case, Ivan has produced a Peltier cooler – a solid-state active heat pump which passes electrical current through the device to transfer heat from one side to another. What's more, he has managed to turn it into something of a work of art. This is a cooler that will look stunning on your desk.

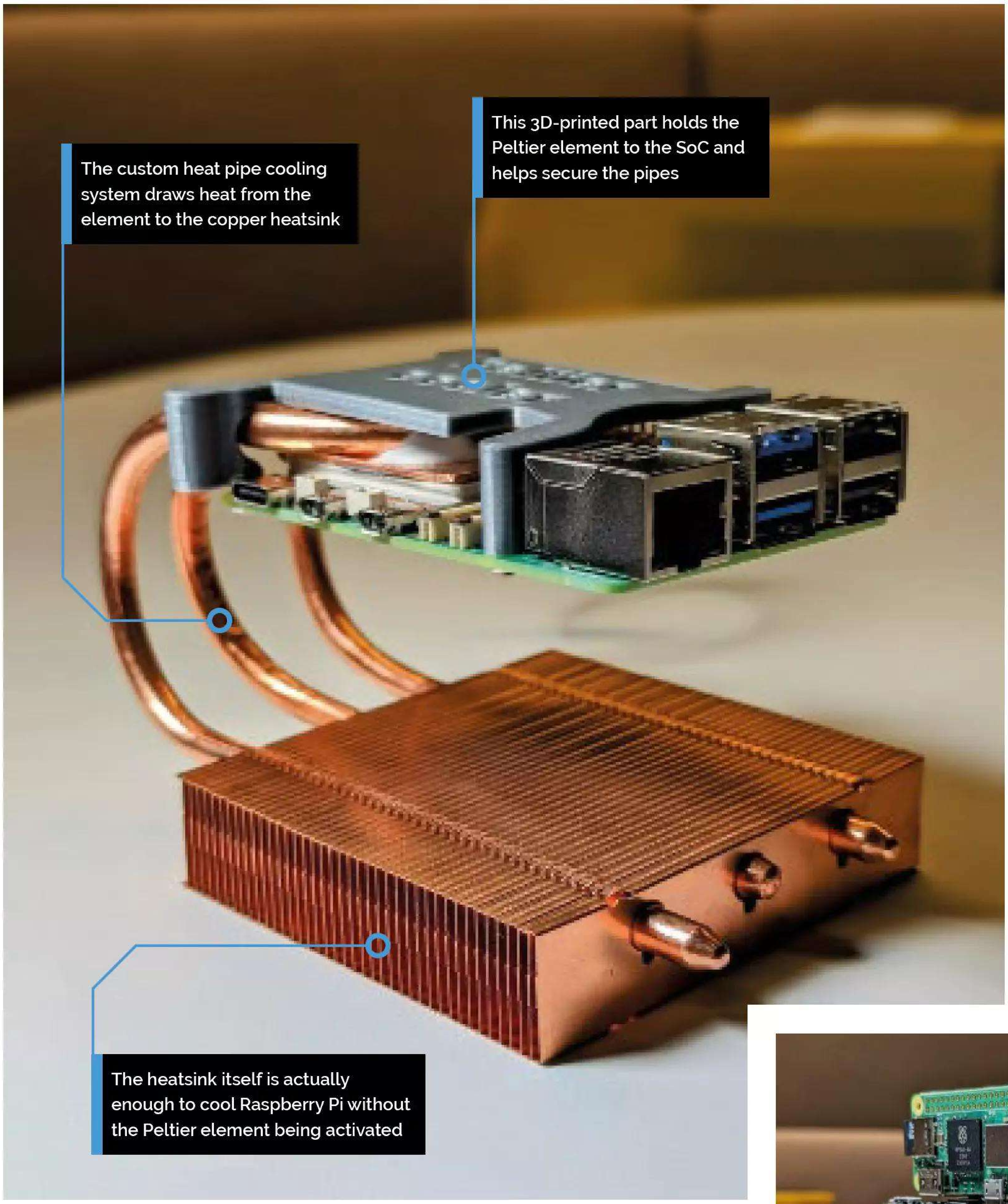
A cool head

Ivan was inspired to create his Peltier cooler following the success of one of his previous projects: a Kubernetes cluster of four Raspberry Pi 4 computers with a similarly unique custom cooling system. He reckoned the Peltier cooler would enable him to overclock Raspberry Pi 5's CPU and ensure it didn't boil while creating something a little more out-of-the-ordinary.

"I originally wanted to make a unique cooling solution for an overclocked Raspberry Pi CPU and I didn't want it to be trivial like water cooling," he says. "I looked at what I had in stock from other projects as the basis for my inspiration and a concept was born. What could be cooler than transferring heat with electricity? I love unusual approaches, and this seemed like a very unusual technology, albeit requiring an awful lot of power."

Ivan surmised that a custom system with thermal tubes and a Peltier element would prove to be the ultimate cooler. "It is potentially possible to reduce temperatures below 0°C although, due to condensation, this is impractical," he says. At the same time, he wanted to create something that would turn heads. "It needed to generate interest and controversy," Ivan adds. "I didn't want a passing project with a banal appearance that people would see in their feeds on their phone and just think, 'okay'".

As such, Ivan was brave. He 'delidded' his Raspberry Pi 5's CPU – a necessary process when looking to take overclocking of a processor to the extreme. It requires separating the metal integrated heat spreader from the top of the CPU and removing the stock thermal paste. It's a bit like brain surgery but, by making use of a high-quality thermal paste or liquid metal instead, it



The custom heat pipe cooling system draws heat from the element to the copper heatsink

This 3D-printed part holds the Peltier element to the SoC and helps secure the pipes

The heatsink itself is actually enough to cool Raspberry Pi without the Peltier element being activated

Quick FACTS

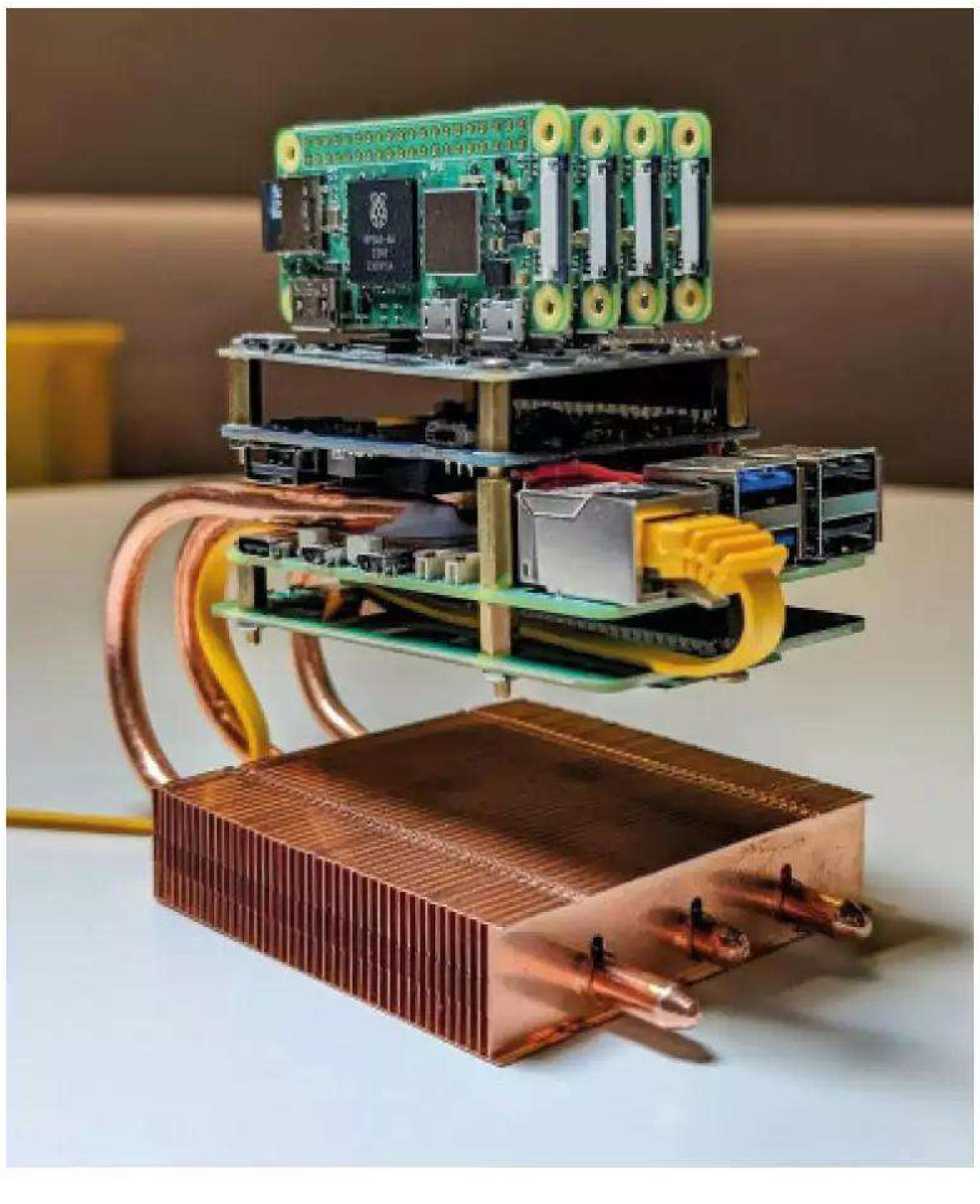
- It uses inexpensive parts from AliExpress
- The project has a Peltier element
- It's possible to reach sub-zero temperatures
- That's the theory. It's not quite reality
- The project has since evolved into a cluster

helps to improve a system's cooling performance. One wrong move, however, and you can wave goodbye to the CPU.

"It's super easy if your hands aren't shaking," Ivan shares. "I used a sturdy kitchen knife which is a great tool but there are several elements on the processor under the cover that can be damaged besides the crystal. You have to be careful."

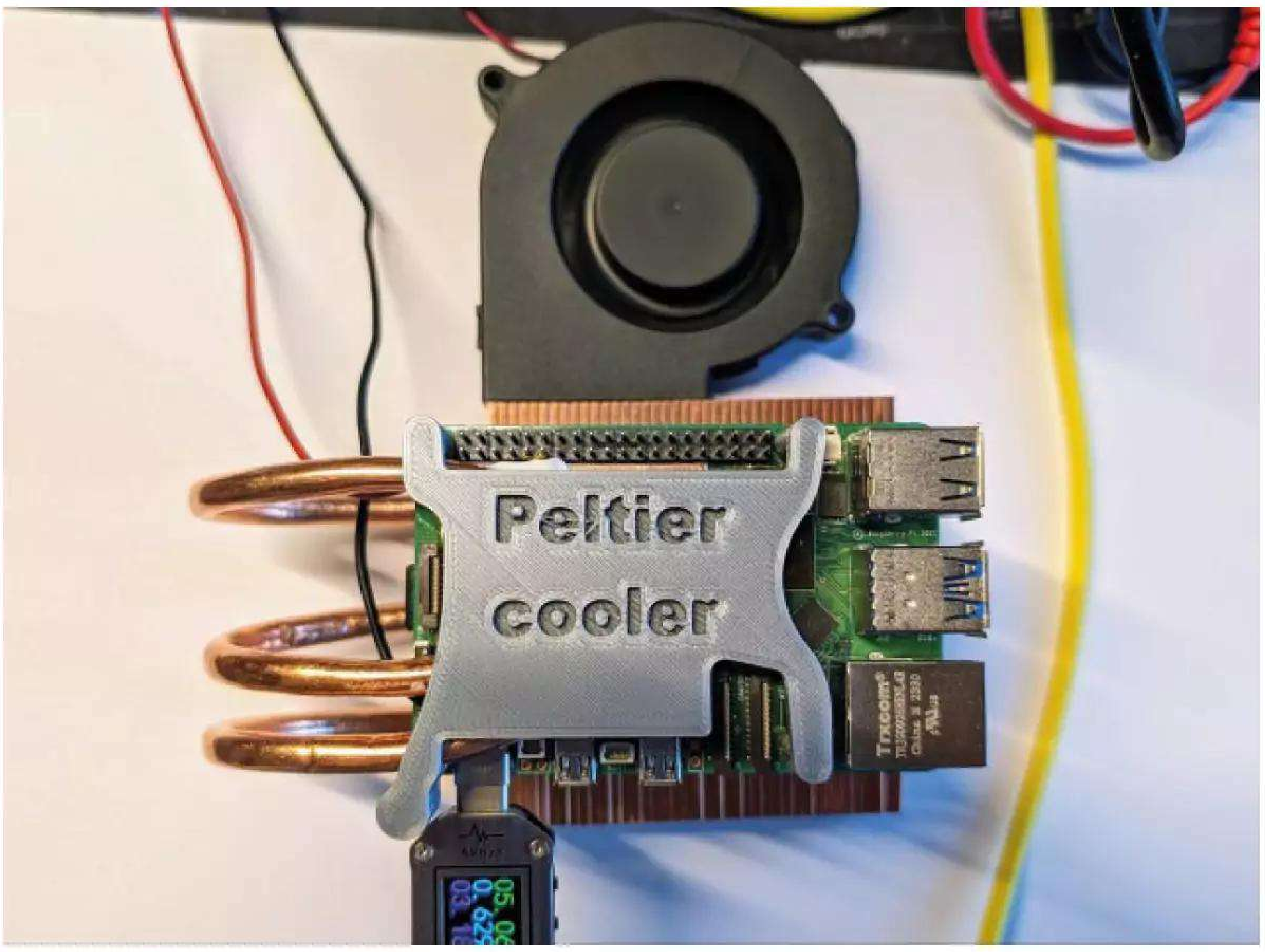
Heat is on

Once the CPU was delidded and Ivan had exposed the Broadcom BCM2712 SoC with its four ARM Cortex-A76 CPU cores, he could move on to the next steps in his project. "I looked at what I

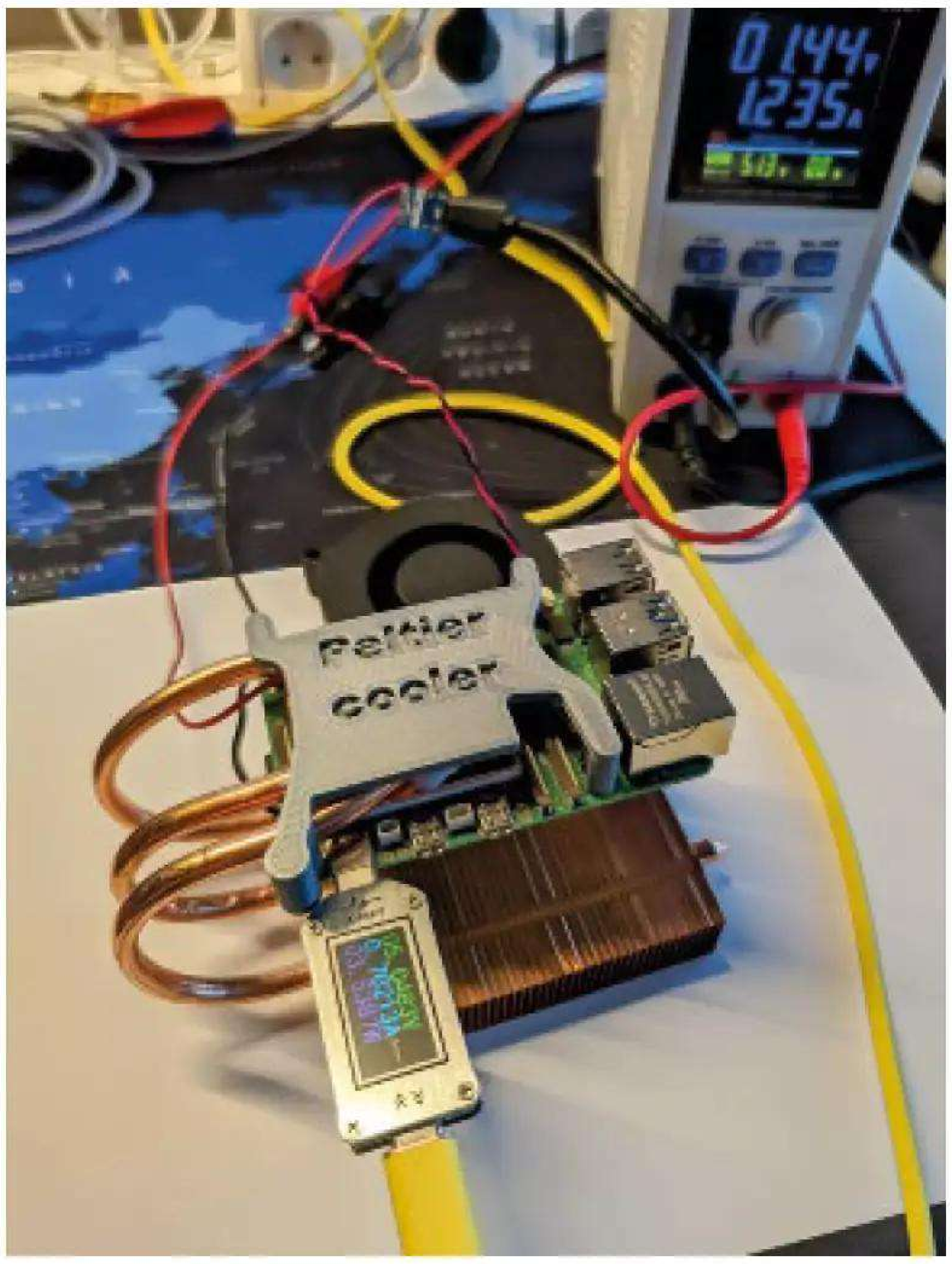


▶ The project has since evolved into a striking non-Peltier, cluster that makes use of four Raspberry Pi Zero 2W boards as a HAT

▼ The simple mount helps to hold everything together and adds a nice finishing touch



▶ Ivan tested the Peltier cooler and found that there was no need to give the Peltier module more energy



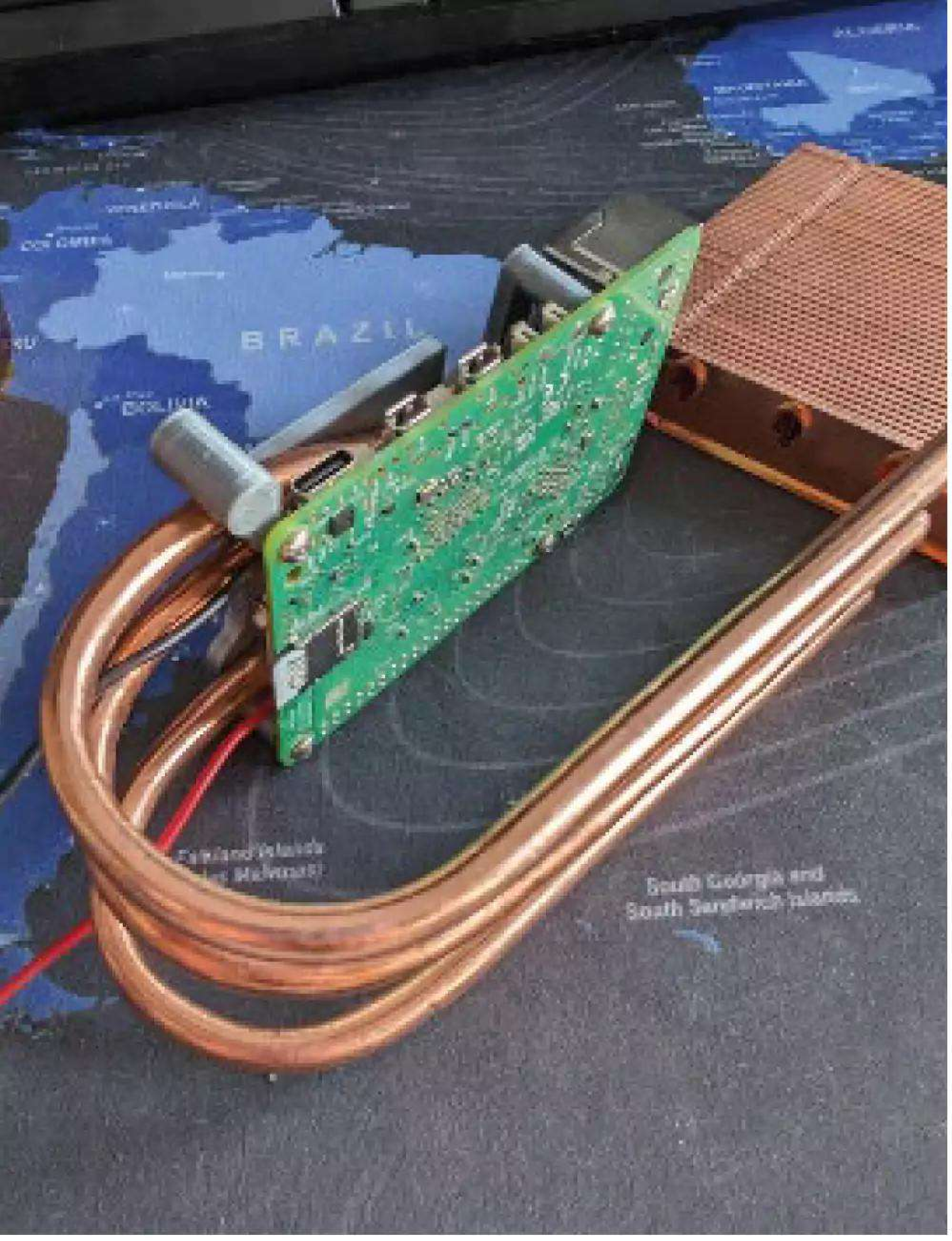
had in my drawer from other projects and envisioned what I could build from that," he says. "There were no sketches, drawings or 3D models beforehand."

Among the pieces he had to hand were copper heatpipes and a copper heatsink, both of which he had previously purchased very inexpensively from AliExpress. Although simple, the pieces also looked striking and Ivan's idea was to connect the thermal tubes to the CPU and pull the heat from the element into the heatsink. To do so, he needed to bend the heatpipes, fasten them with thermally conductive glue and use the heatsink as a base.

"The unconventional positioning of the radiator at the bottom has been criticised and there have been more than a few questions and comments saying this solution won't work," he says. "But it works fine. It doesn't really matter how and where the heat pipes go. They transfer heat from the hot section to the cold section, equalising their temperature. Simply put, there is a liquid inside at low pressure and a spongy structure which allows the capillary effect to transfer heat very efficiently. It's orders of magnitude more efficient than just a piece of copper or aluminium."

Clock ticking

When the project was complete, however, Ivan made a discovery which surprised him. Having spent many hours testing, he simply couldn't find a way of overclocking Raspberry Pi 5's CPU beyond 3GHz. "Overclocking is limited unless there is a way to get around this limitation," he laments. "I


BRAZIL
BOLIVIA
Falkland Islands (Islas Malvinas)
South Georgia and South Sandwich Islands

Although buying an official Active Cooler is easier, there's no denying that Ivan's own cooler looks, well, cooler

really thought that extreme overclocking would require true extreme cooling. On Raspberry Pi's Compute Model 4 IO Board, though, [which has a BCM2711 quad-core Cortex-A72 SoC running at 1.5GHz], I was able to achieve 3GHz at temperatures around 0°C."

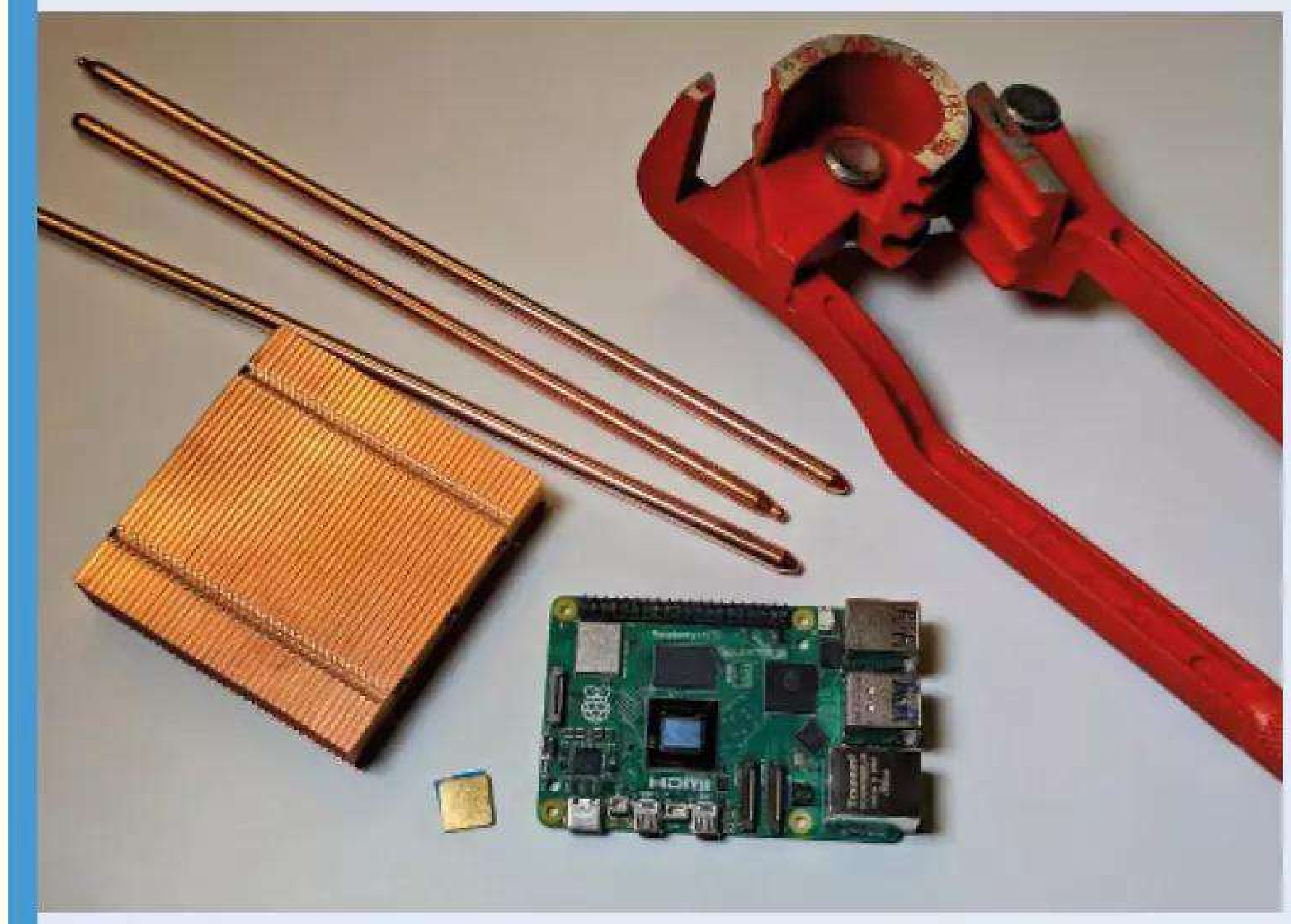
In that sense, the project doesn't quite achieve its intended aim of keeping a stretched Raspberry Pi cool. "The Peltier element generates a lot of heat, much more than my Raspberry Pi computer," Ivan adds. "To power it for minus-Celsius temperatures, you need probably about 50 to 70W of electricity and a small heatsink can't cope with it."

Yet it has been a success overall. "In quiet mode, about 15W, the Peltier element works well and keeps Raspberry Pi's CPU around 40°C under full load," he says. Even so, there are limitations. "The heatsink is very much blown out, because the heat from Raspberry Pi from the Peltier element goes to it." Rather than become disheartened, though, Ivan has moved the project on. He's added four Raspberry Pi Zero 2W boards as a HAT, as well as an NVMe SSD and introduced custom passive cooling.

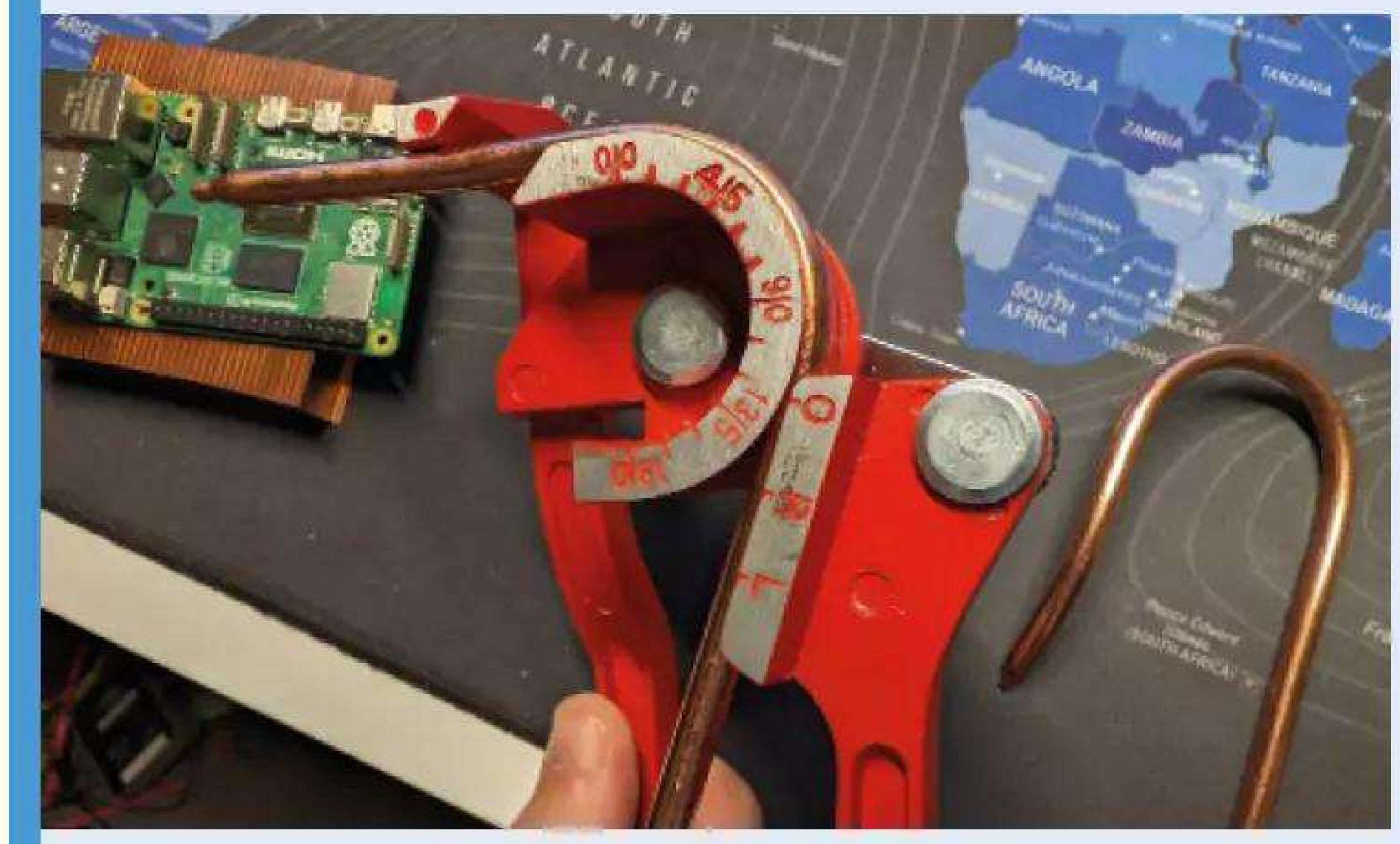
> " An interesting experience, but it doesn't make much sense in real life "

"In short, it's been an interesting experience but it doesn't make much sense in real life, especially because of the limitations of overclocking which is why I removed the Peltier element and converted the project to a new cluster," he says. Even so, in either guise, it still looks very cool and that, in some respects, also matters greatly.

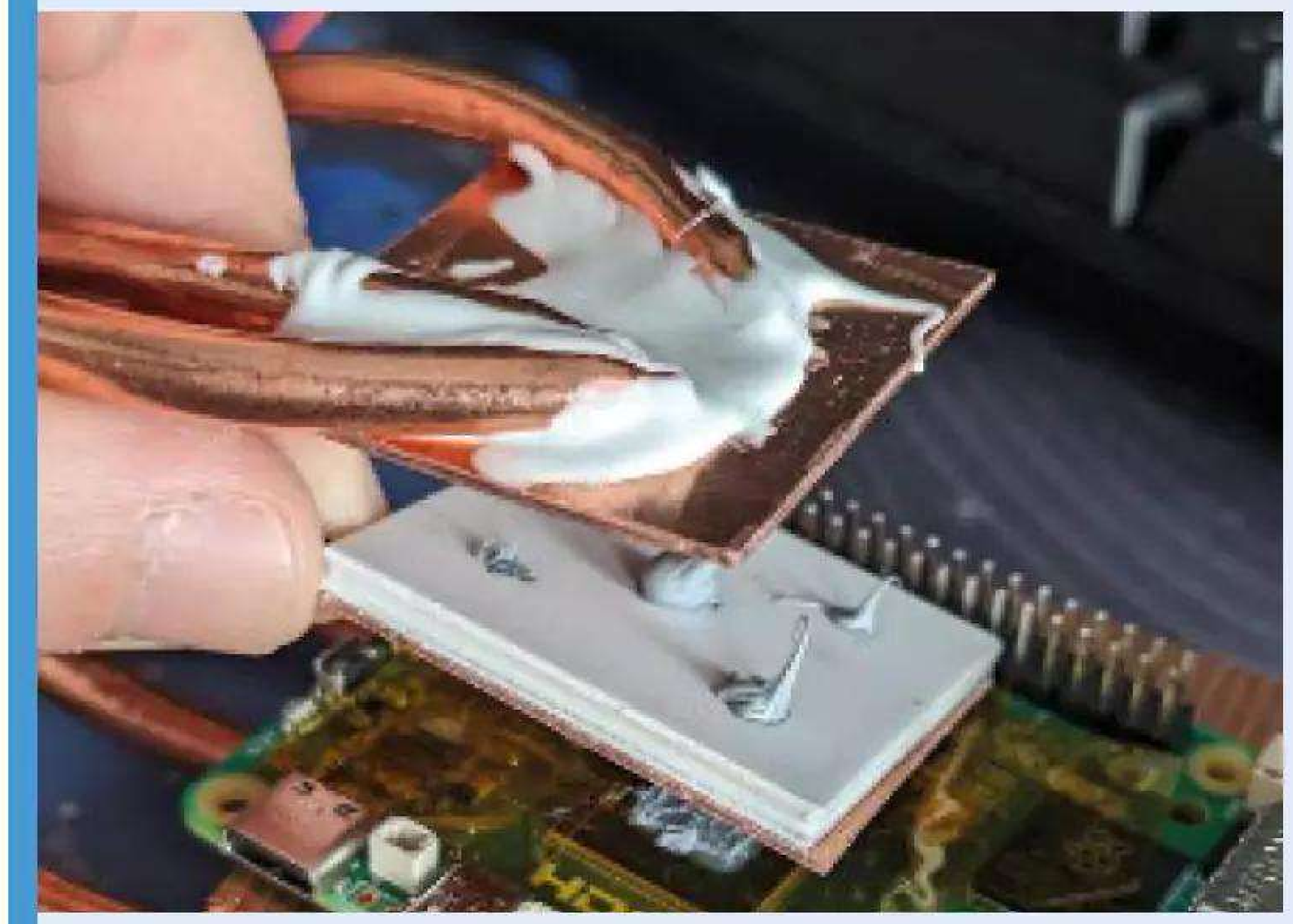
Building the cooler



01 Ivan already had all of the parts he needed since he'd previously bought them for other projects. It was then a case of figuring how to connect them so that the project would be eye-catching and effective.



02 The thermal pipes needed to be bent so that they would not only curve outwards and connect the Raspberry Pi computer to the heatsink, but also bend inwards so that all three ends would be positioned over the CPU.



03 The pipes were connected to the Peltier element – two pieces of copper with a conductive filling. One side is hot and connects to the heatsink, the other is cool and connects to the SoC. Heat is then able to be drawn away.

SUCCESS STORY magpi.cc/success

Raspberry Shake seismometers

Raspberry Shake set out to build a professional-grade seismograph that's also accessible to citizen scientists. Now with thousands of installations around the globe, this highly sensitive Raspberry Pi-based instrument is powering cutting-edge research, writes Rosie Hattersley

Raspberry Shake is a sophisticated seismograph, widely used today in professional scientific research as well as in the citizen science projects in which it has its roots. Designed and built by founder Branden Christensen and an enthusiastic team of Panama-based seismologists and geophysicists, its first outing was to measure and monitor tremors on the slopes of Volcán Barú – the 3474m volcano that is Panama's highest peak. Christensen has been designing tools to measure seismic activity since 2011, basing early prototypes around Technologic's single-board computer, one of the few such products on the market at the time. However, the project that grew into Raspberry Shake needed something leaner and far less expensive in order to develop a scalable, affordable product.

The challenge

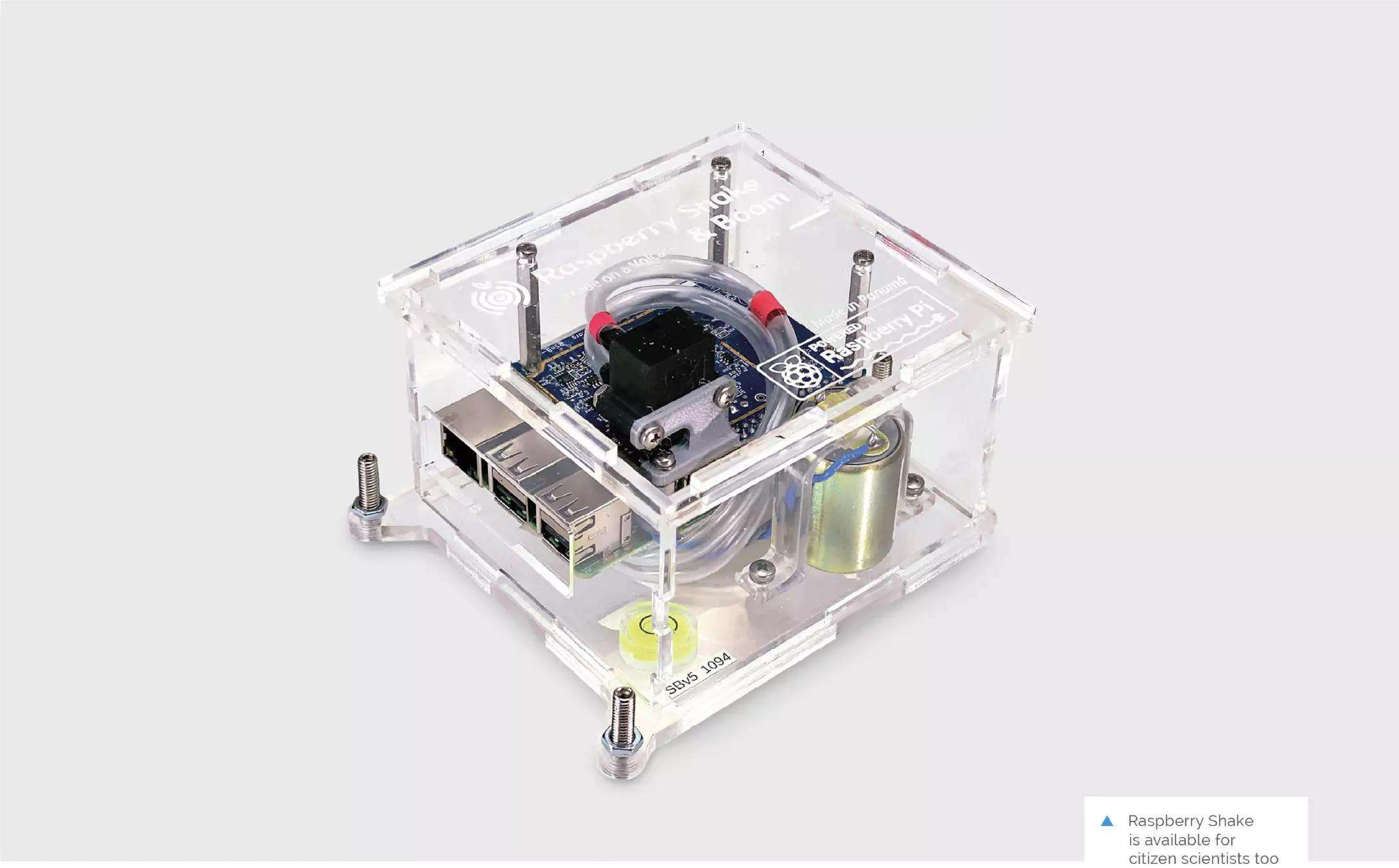
Experimental designs with a variety of SBCs and seismographs took place between 2012–16, and the company's first crowdfunding campaign launched with a different embedded single-board computer, but Raspberry Pi had already begun to look like one of the strongest hardware candidates. "Each time the project failed to attract investment and interest at the crowdfunding stage. All these different iterations failed in the sense that they didn't create market share," explains Christensen. They hoped the runaway success of Raspberry Pi might make the difference.

The solution

Raspberry Shake tested several SBCs, including Beaglebone Black and the original Raspberry Pi 1 Model B, using the latter with an Ethernet connection to provide a link to a data dashboard. "We did the calculations and determined that we could use this less expensive computer," says Christensen. "So we built one of our first independently sourced and designed seismographs based on Raspberry Pi 2B."

Using the Raspberry Pi "was relatively inexpensive for what it could do", while "in terms

▶ Raspberry Shake sits in a clear acrylic box



▲ Raspberry Shake is available for citizen scientists too

of the design and operation of the [seismo]graph, the technical requirements of the CPU, RAM and hard disk space were minimal".

Redesigning the seismograph around Raspberry Pi and promoting the hardware inside via a rebrand, Raspberry Shake's initial Kickstarter campaign set a goal of attracting a few dozen backers, but quickly ended up with hundreds. The project had struck a chord with citizen scientists keen to contribute to crowdsourced data, detecting and recording earthquakes from home using a professional-grade seismograph.

Why Raspberry Pi?

With a plan to use crowdfunding to get the first product off the ground, Raspberry Shake hoped to gain support among the extensive and growing community of people using Raspberry Pi hardware. "We strategically chose the Raspberry Pi computer because we had been watching everything that was going on with the single-card computer industry and we saw that Raspberry Pi was widely adopted." The team reasoned that creating a seismographic board that could attach to Raspberry Pi easily would mean that someone who had a Raspberry Pi board already might repurpose it and use it to join Raspberry Shake's growing seismograph community.

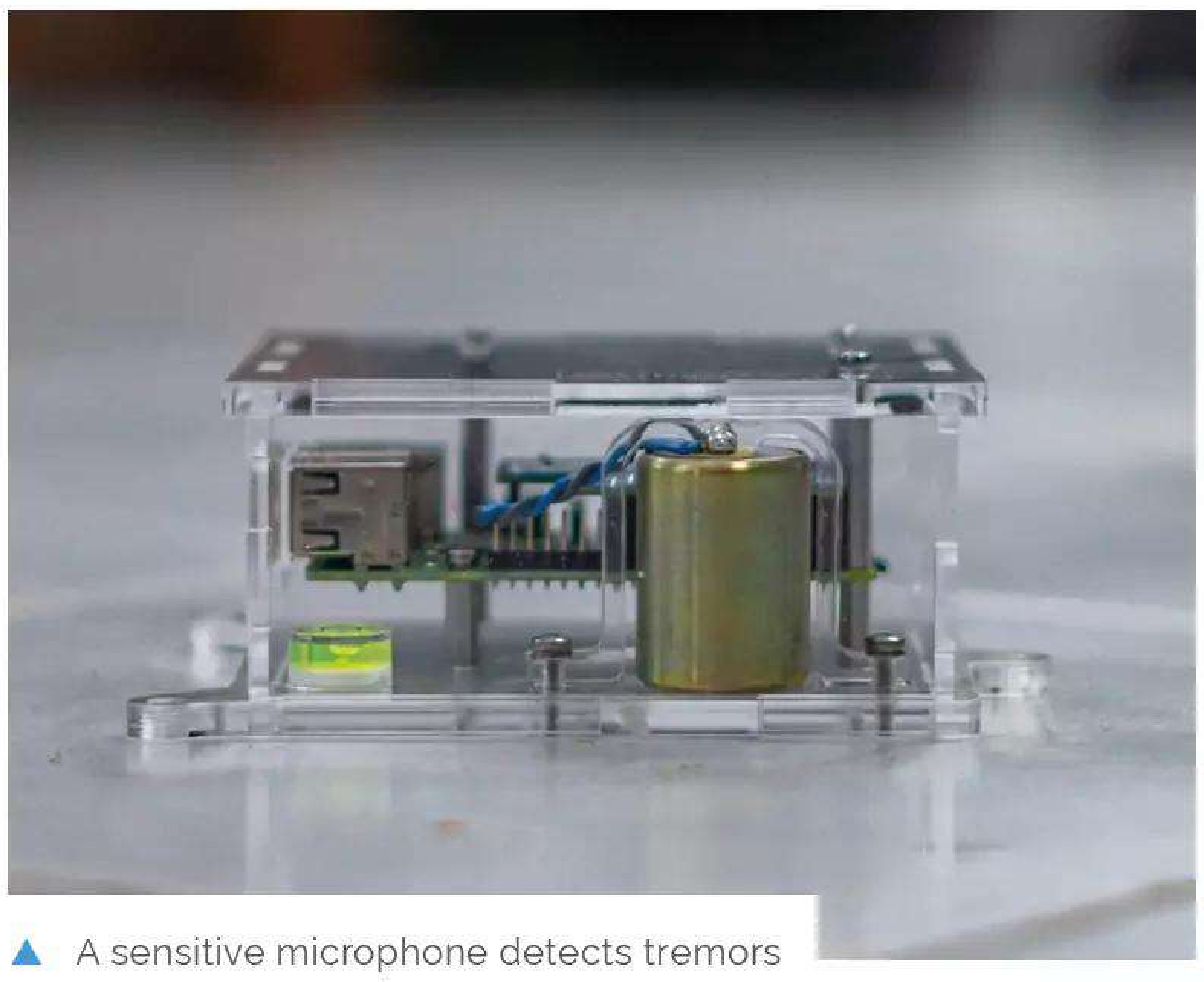
The project "really took off" in the first weeks of the campaign, says Christensen. "We hit our funding target… and then we went way beyond that!" he recalls, describing their elation when they exceeded their initial target by a factor of 14.

An unexpectedly successful crowdfunding campaign meant, of course, that the team had far more orders to fulfil than they had envisaged. Having previously manufactured just a few hundred seismographs in a six-month period, they suddenly needed to source parts for and produce 800 all at once. Panama's under-developed infrastructure in terms of international shipping was a challenge, too, now that they needed to ship to more than 70 countries. Happily, Panama's Ministry of Commerce was very supportive: "They created a programme for us to ship inexpensively through the Panamanian postal service, since there were no DHL or FedEx options back then," says Christensen.

Raspberry Shake would go on to find some extremely reliable manufacturing partners that they continue to work with, as well as establishing a really useful working relationship with Raspberry

Pi. Christensen attributes some of the excitement around the product to close communication with Raspberry Pi and its strong UK customer base. As well as providing invaluable early exposure when Raspberry Shake launched, Raspberry Pi was supportive when Raspberry Shake faced some tricky shipping issues. “For all the bumps, they were right there with us, which was really nice. So they supported us and, within months, we resolved all the issues.”

Six months after the crowdfunding campaign, all backers had received their Shakes, and “there was a citizen science network of between 500 and 600 people online streaming real-time data to us”.


▲ A sensitive microphone detects tremors

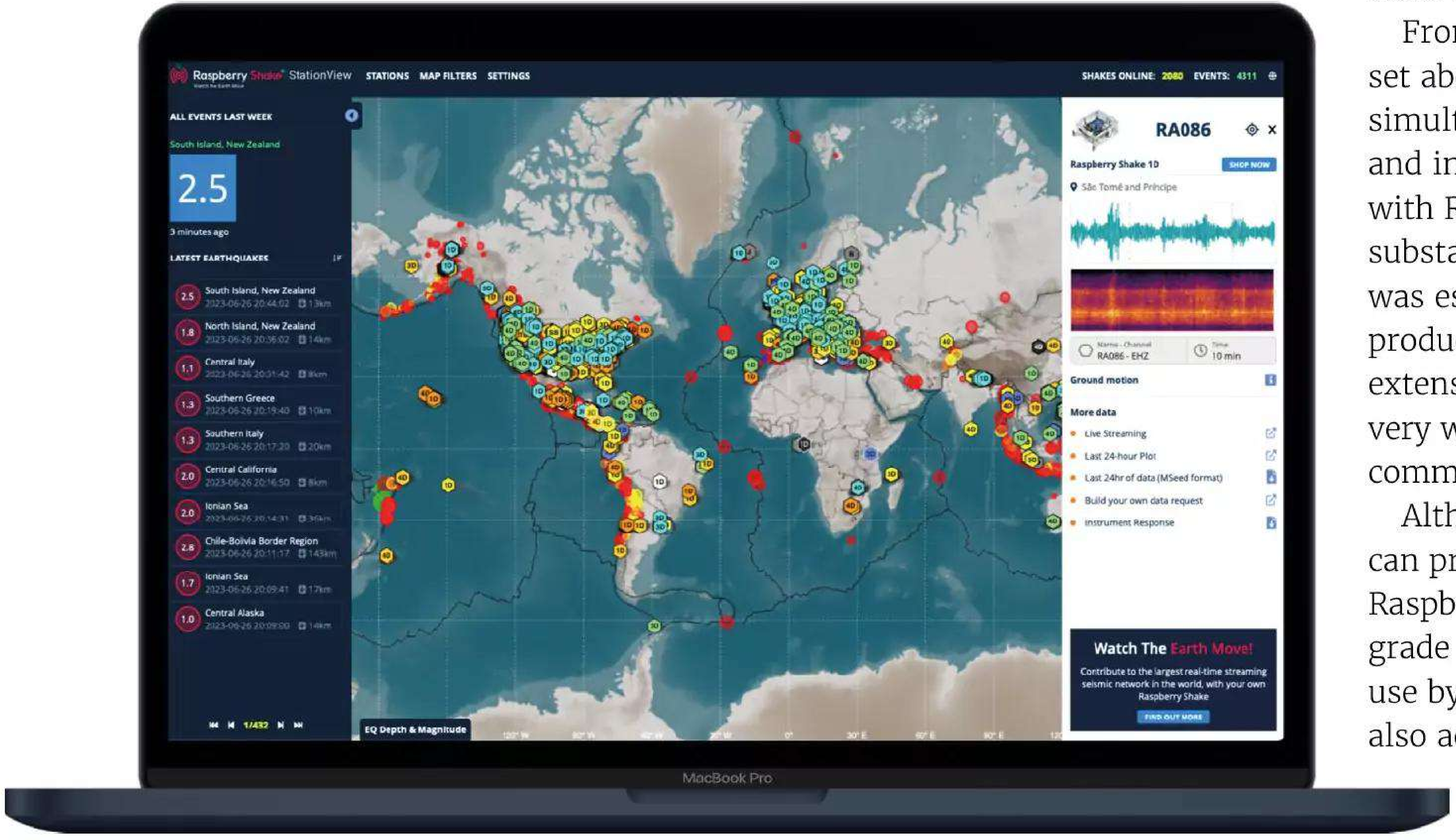
The results

Once backers received their seismometers, they swiftly set them up and shared their findings online. An accidental deviation from the spec gave a boost to early engagement: the team was initially puzzled to find hundreds of reports of seismic activity from their major customer base in the UK, given the country’s lack of earthquakes. The devices were detecting “very, very low magnitude Earth motion that would come from really distant earthquakes, literally half a world away.” Further investigation revealed that a professional-grade sensor, specified for a research version of Raspberry Shake designed for volcano monitoring at the US Geological Survey, had accidentally been included in the model that went out to crowdfunder backers. The unexpectedly high sensitivity of the instrument generated a lot of excitement, with most early customers discovering they were able to detect earthquakes despite being thousands of miles from areas of significant seismic activity.

From this point on, Raspberry Shake set about tailoring the design to appeal simultaneously to professional geologists and interested individuals, sticking with Raspberry Pi both because it had substantial market share and because it was established as a relatively mature product. Another big plus was its extensive community, which worked very well for Raspberry Shake’s own community adoption ambitions.

Although there exist seismographs that can provide readings at a lower cost, Raspberry Shake is a professional-grade seismograph suitable for use by geologists that, crucially, is also accessible to a substantial and geographically widespread community of citizen scientists

▼ You can see global data from the network of Raspberry Shakes


Raspberry Shake StationView
STATIONS
MAP FILTERS
SETTINGS
SHAKES ONLINE: 2080
EVENTS: 4311
ALL EVENTS LAST WEEK
South Island, New Zealand
2.5
3 minutes ago
LATEST EARTHQUAKES
2.5 South Island, New Zealand 2023-06-26 20:44:02 13km
1.8 North Island, New Zealand 2023-06-26 20:36:02 14km
1.1 Central Italy 2023-06-26 20:31:42 8km
1.3 Southern Greece 2023-06-26 20:19:40 10km
1.3 Southern Italy 2023-06-26 20:17:20 20km
2.0 Central California 2023-06-26 20:16:50 8km
2.0 Ionian Sea 2023-06-26 20:14:31
2.8 Chile-Bolivia Border Region 2023-06-26 20:11:17 143km
1.7 Ionian Sea 2023-06-26 20:09:41 17km
1.0 Central Alaska 2023-06-26 20:09:00 14km
1/432
EQ Depth & Magnitude
RA086
Raspberry Shake 1D
SHOP NOW
São Tomé and Príncipe
RA086 - EHZ
10 min
Ground motion
More data
Live Streaming
Last 24-hour Plot
Last 24hr of data (MSeed format)
Build your own data request
Instrument Response
Watch The Earth Move!
Contribute to the largest real-time streaming seismic network in the world, with your own Raspberry Shake
FIND OUT MORE
MacBook Pro

▲ There are several versions depending upon your needs

> Raspberry Shake is a professional-grade seismograph suitable for use by geologists

in educational settings and at home. Measurements collected in the latter setting constitute a rich dataset that has proven immensely valuable to professional earth sciences researchers.

Raspberry Shake's product line now consists of several different models targeting different applications and price points, with devices that variously monitor up-and-down motion, lateral motion, and acoustic atmospheric and subterranean disturbances; there's also a dedicated ShakeNet app. All of the instruments send data to a cloud service, and it is viewable on the Raspberry Shake website by event type, location, intensity, and duration. It's these measurements recorded by the Raspberry Shake Data Center that are used by seismologists around the world for scientific research, and widely cited in academic papers and publications.

Raspberry Shake data has contributed to research into the 2021 Haiti earthquake, icequakes in Antarctica, and traffic volume in heavily urbanised areas in the UK, to mention just three of many diverse applications. More recently, the technology is being deployed alongside another Raspberry Pi-based instrument, a 3D timelapse device installed in volatile areas in the Alps to detect early warnings of the avalanches that have cost multiple skiers' lives over the past few years.

SUBSCRIBE TODAY FOR JUST £10

Get 3 issues + FREE Pico W

Subscriber Benefits

- FREE Delivery
  Get it fast and for FREE
- Exclusive Offers
  Great gifts, offers, and discounts
- Great Savings
  Save up to 23% compared to stores

Subscribe for £10

- Free Pico W
- 3 issues of The MagPi
- £10 (UK only)

Subscribe for 6 Months

- Free Pico W
- 6 issues of The MagPi
- £30 (UK) $43 (USA)
  €43 (EU) £45 (Rest of World)

☎ Subscribe by phone: **01293 312193**

Subscribe online: **magpi.cc/subscribe**

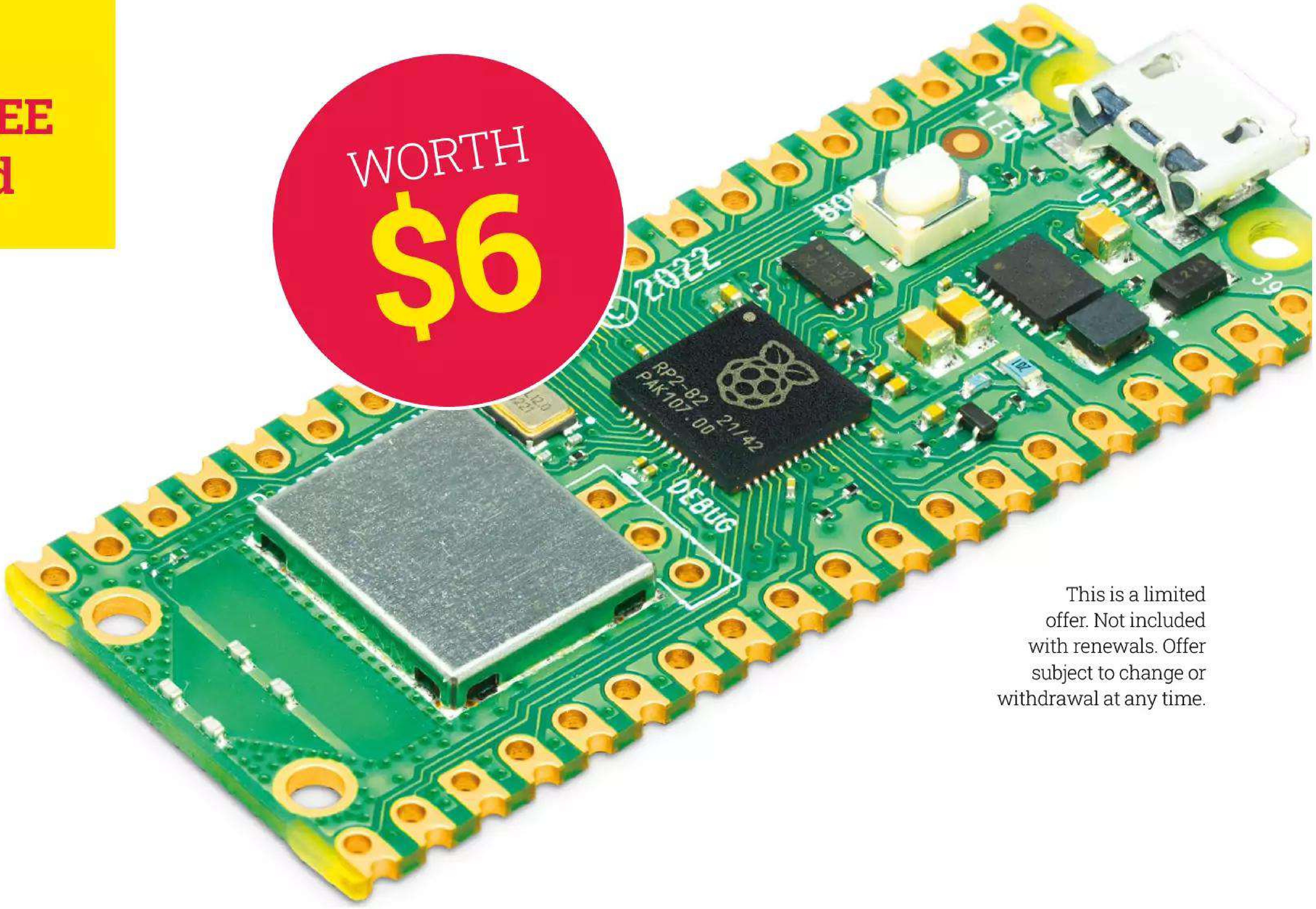
Email: **magpi@subscriptionhelpline.co.uk**

Subscribe for £10 is a UK-only offer. The subscription will renew at £15 every three months unless cancelled. A free Pico W is included with a 6-month subscription in USA, Europe and Rest of World.

SUBSCRIBE TODAY AND GET A

FREE Raspberry Pi Pico W

Subscribe in print today and get a FREE development board



WORTH $6

- A brand new RP2040-based Raspberry Pi Pico W development board
- Learn to code with electronics and build your own projects
- Make your own home automation projects, handheld consoles, tiny robots, and much, much more

This is a limited offer. Not included with renewals. Offer subject to change or withdrawal at any time.

YOUR OFFICIAL RASPBERRY PI MAGAZINE
The MagPi
TROUBLESHOOTING GUIDE
UPCYCLE OLD SONOS SYSTEMS WITH RASPBERRY PI
NEXT GEN RETRO

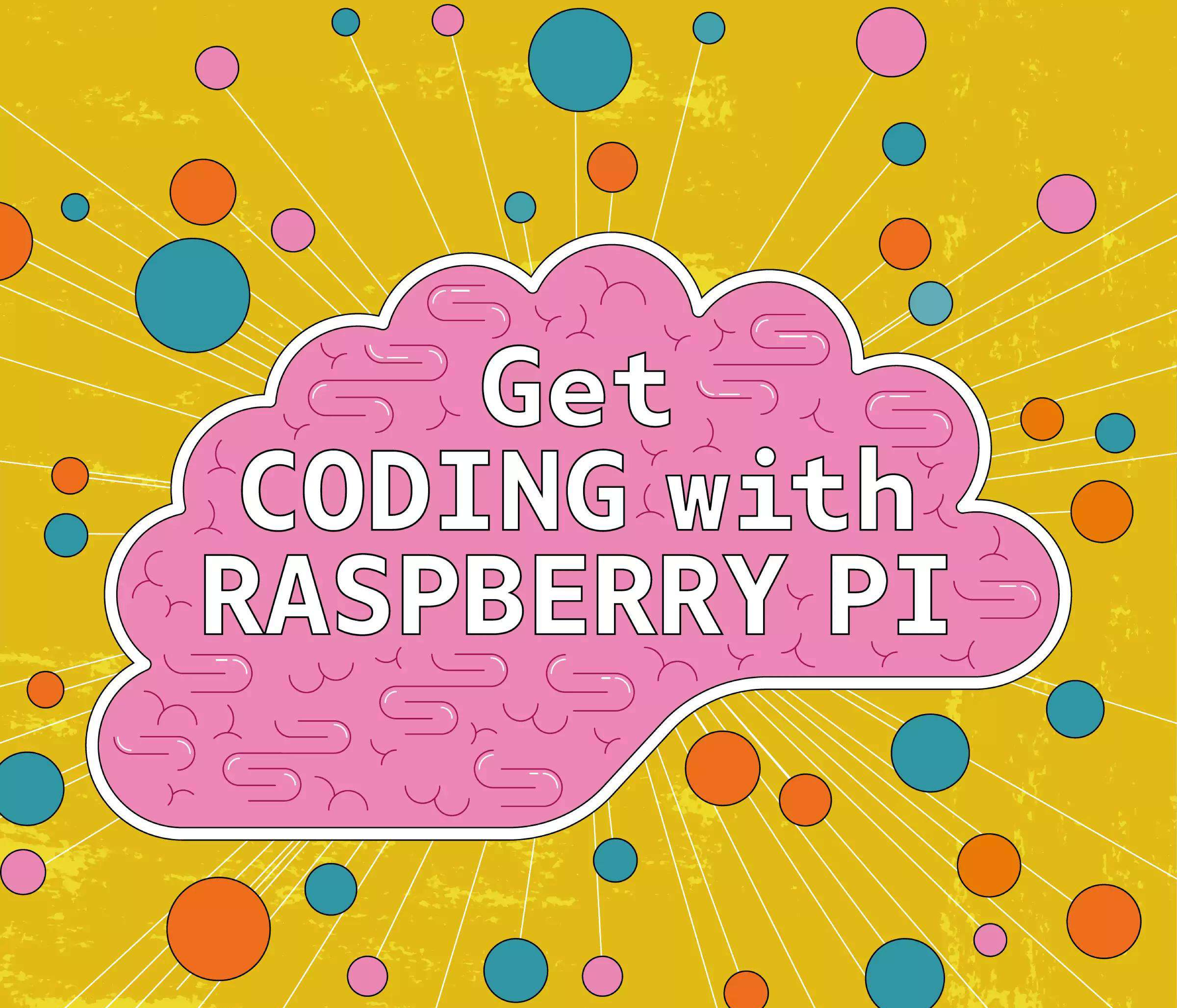
Buy now: magpi.cc/subscribe

SUBSCRIBE on app stores

From **£2.29**

Available on the App Store

GET IT ON Google Play

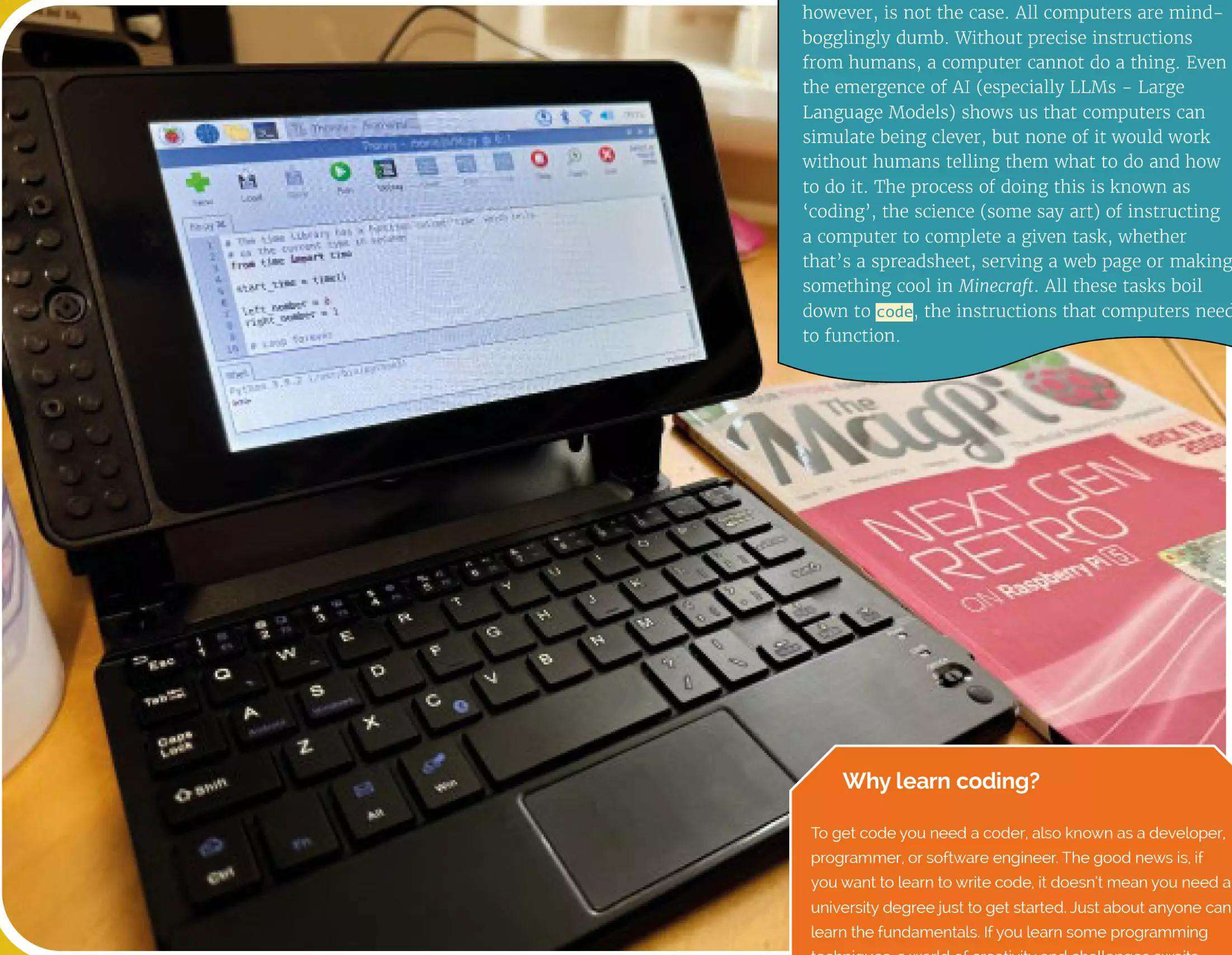


Get CODING with RASPBERRY PI

Coding is one of the most rewarding things you can do with Raspberry Pi. But where to start? **PJ Evans** takes to the keyboard

> “All computers are mind-bogglingly dumb. Without precise instructions from humans, a computer cannot do a thing”

People often think of computers as extremely clever things, maybe much more so than the average human. After all, they can think faster than we can and computers are knockout at maths. They can remember more than us and always have perfect recall. This, however, is not the case. All computers are mind-bogglingly dumb. Without precise instructions from humans, a computer cannot do a thing. Even the emergence of AI (especially LLMs – Large Language Models) shows us that computers can simulate being clever, but none of it would work without humans telling them what to do and how to do it. The process of doing this is known as ‘coding’, the science (some say art) of instructing a computer to complete a given task, whether that’s a spreadsheet, serving a web page or making something cool in *Minecraft*. All these tasks boil down to `code`, the instructions that computers need to function.



```
from time import time

start_time = time()

left_number = 0
right_number = 1

# Loop forever
```

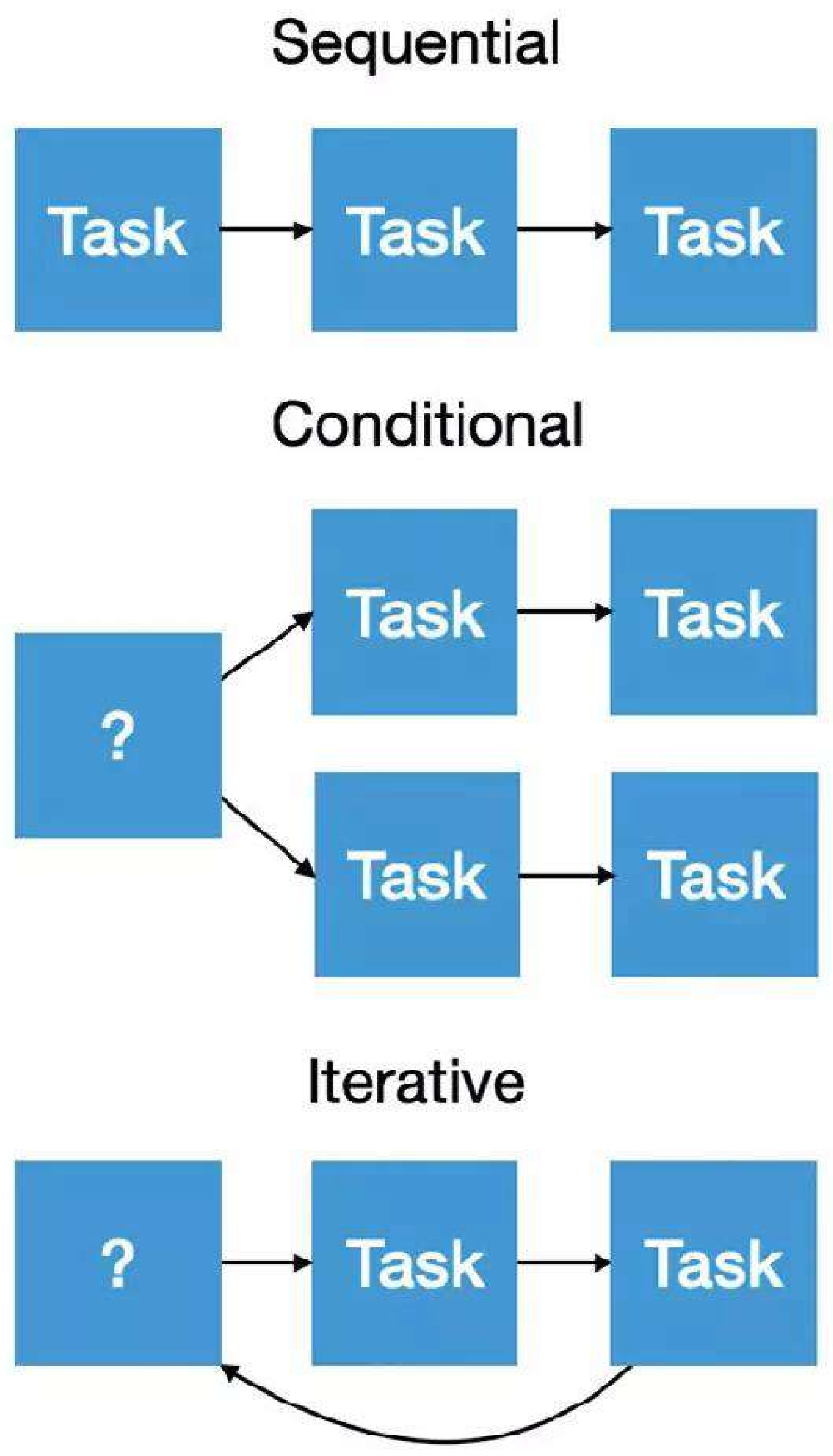
Why learn coding?

To get code you need a coder, also known as a developer, programmer, or software engineer. The good news is, if you want to learn to write code, it doesn’t mean you need a university degree just to get started. Just about anyone can learn the fundamentals. If you learn some programming techniques, a world of creativity and challenges awaits. From automating common tasks, and interacting with the environment to making websites or even games, coding is an essential skill for getting the most out of your computer. Best of all (for the most part) it is a lot of fun and extremely rewarding when you run your program and it works.

Pick your language

With a few exceptions that we'll cover later, nearly all coding is done using text-based programming languages. There are many, many different languages out there, and we'll cover some of the most popular ones here. Why isn't there a single language? Two reasons: firstly, some languages are designed for specific tasks, such as R for data analysis, while others are known as general-purpose languages. Secondly, the people who design these languages rarely agree on anything and are forever trying to improve things by creating new ones.

Languages are typically text-based as that's the easiest way to get information into a computer. All languages have three basic structures: sequential, conditional and iterative. Let's look at these by writing a programme (code that does a specific task) that makes a cup of tea.



▶ The three fundamental types of code structure

Sequential instructions

Sequential instructions are carried out in sequence, one after another. Here are some instructions for making a cup of tea:

1. Fill kettle
2. Switch kettle on
3. Get mug, teabag and milk
4. Place teabag in mug with water
5. Add milk

However, a computer would struggle with this. Firstly it would fill the kettle whether it was already full or not. So let's add a conditional; code that makes a decision:

- If kettle is not full of water then fill the kettle to the top

Now we don't have water all over the kitchen floor. Another problem is that the computer will not wait for the kettle to boil, so we have cold tea! Let's iterate, or continuously run some code until a condition is met:

- While the kettle has not boiled, wait for a bit and check again

Finally, if we wanted many cups of tea we could create a function, or group of commands, called "make tea" and then iterate over it. Functions are especially useful if you need to repeat a task again and again. Need to cater for a big party? For 5,000 times: Run "make tea".

Python

Did you know that the ‘Pi’ in Raspberry Pi is a nod to the Python programming language? Python started life as a way to make code look close to human language and do away with all the weird punctuation of other mainstream languages such as C++.

As a result, Python is undoubtedly the most popular language for beginners. It has a friendly syntax (the term for a language’s rules) and can be extended with modules that make things such as talking to the Raspberry Pi’s GPIO as easy as possible. These modules have led to Python becoming hugely influential in data analysis and artificial intelligence.

Python is a creative wonderland. Interact with the real world using its GPIO tools, build games with the PyGame library, make web applications and more. It’s no coincidence that most coding articles in this magazine are Python-based.

Python comes as standard with all Raspberry Pi operating systems, along with Thonny, an application (known as an IDE) that helps you interact with the language.

Let’s create our first program. Open up Thonny (Raspberry Pi menu > Programming > Thonny IDE and enter the two lines of code from **hello_world.py**. Note that when you press **RETURN** after the first line Thonny will automatically indent the second line (we will explain all this in a tutorial later in the mag). Just know for now that the indent is four spaces. Check every single character is written exactly as you see it in the code listing.

Click Run, and you will see “Hello World” output five times in the Shell below. Congratulations! You’ve written a computer program.

Check out Scratch

MIT’s Scratch introduces the concepts of coding in as friendly a way as possible. Instead of text, you can drag and drop colourful blocks to create simple programs and fun games. It’s the perfect first programming language and has been a part of Raspberry Pi OS since the beginning.

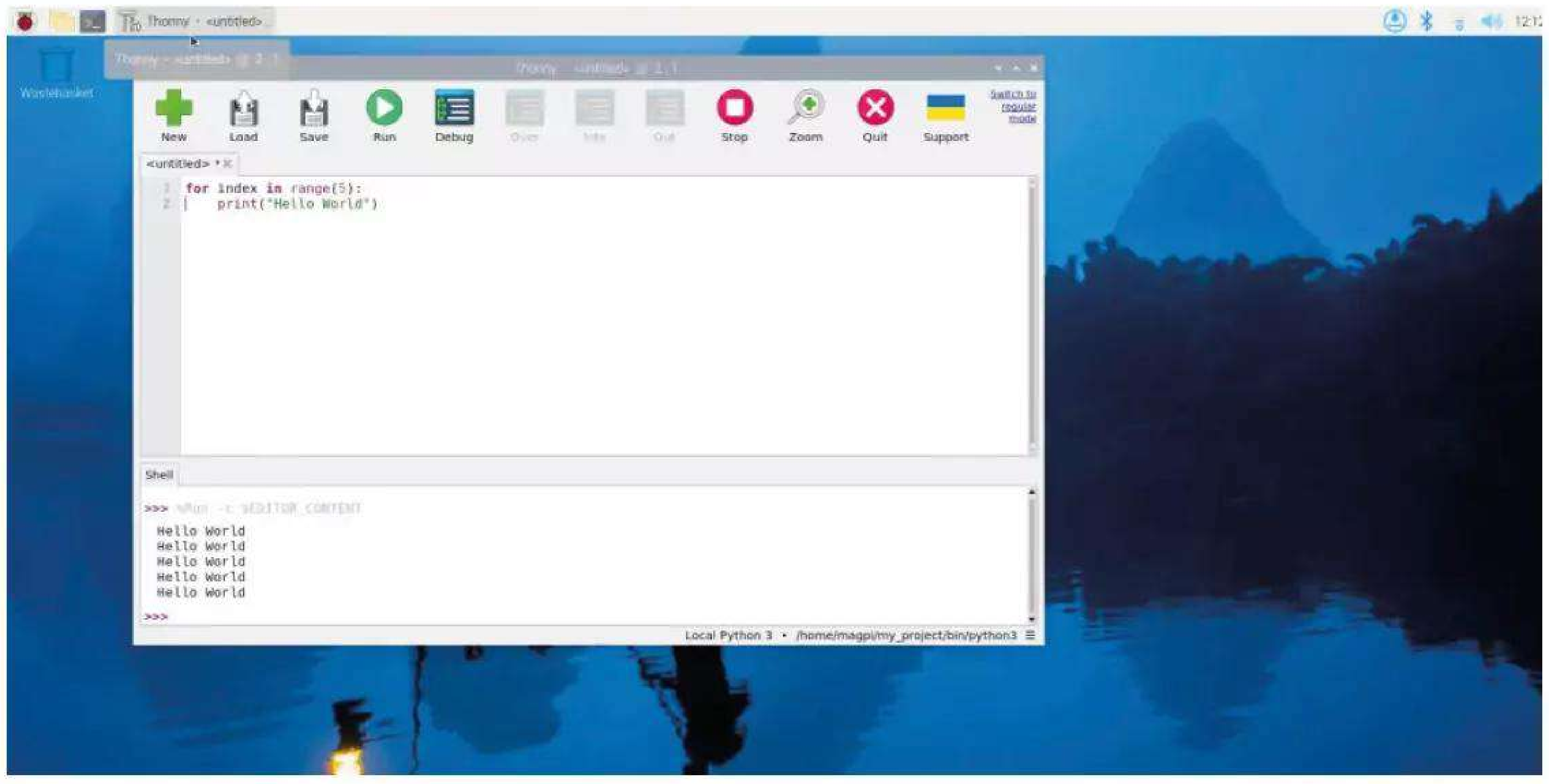
scratch.mit.edu.

> “ Python is undoubtedly the most popular language for beginners ”

hello.py

▶ Language: **Python**

```
for index in range(5):
    print(“Hello world!”)
```



```
for index in range(5):
    print("Hello World")
```

```
>>> %Run -c $EDITOR_CONTENT
 Hello World
 Hello World
 Hello World
 Hello World
 Hello World
>>>
```

Local Python 3 • /home/magpi/my_project/bin/python3

Node.js

JavaScript is one of the most popular programming languages in the world. It started life in web browsers and is the language used to make websites interactive. If you visit a website, chances are your browser runs JavaScript to make it work. Node.js is a system for running JavaScript locally, rather than in the web browser. Node apps power the back-end of web services all over the world.

It's less friendly than Python with a syntax closely resembling C++, but it's also a lot faster. Node.js appeals to developers as it balances speed and complexity well. Like Python, Node.js has modules that can add all kinds of capabilities to the language.

So, if you're interested in writing APIs (Application Programming Interfaces – used by web browsers to send and receive data directly from internet servers) or websites, Node.js is where you need to be looking. It's now well supported by Raspberry Pi and is blisteringly fast on the new Raspberry Pi 5.

hello.js

Language: **Node.js**

```
for(let index = 0; index < 5; index++) {
   console.log('Hello world!');
}
```

Go

A newer player in the field, Go was developed by Google engineers fed up with inconsistencies between different languages (programmers never agree on these things). Python and Javascript are run-time interpreted, which means your code is compiled as it runs. This has many advantages, especially for developers, but it does cause significant overhead and makes things run slower.

Go can be both run-time interpreted and compiled, which means its code can be turned into machine-level code, which gives a huge speed increase.

For a more complex language, Go is friendly to use and easier to learn than most of its peers. It can be used for pretty much any purpose and is well supported on Raspberry Pi. Google has focussed on a clear syntax that combines the brevity of JavaScript with the readability of Python. Go is now starting to eat into Node's dominance of the web server world, so the language is well worth your attention.

hello.go

Language: **Go**

```
package main
import "fmt"
func main() {
   for index := 0; index < 5; index++ {
      fmt.Println("Hello world!")
   }
}
```

Coding tools and tips

You can write code in just about any text editor, or even straight into documents using the terminal. However, you'll probably want to use a dedicated text editor or IDE (Integrated Development Environment).

IDEs are word processor-style apps that help you code by detecting bugs and syntax errors, and show help on command structures. They often have plugins to help you specifically with your language of choice.

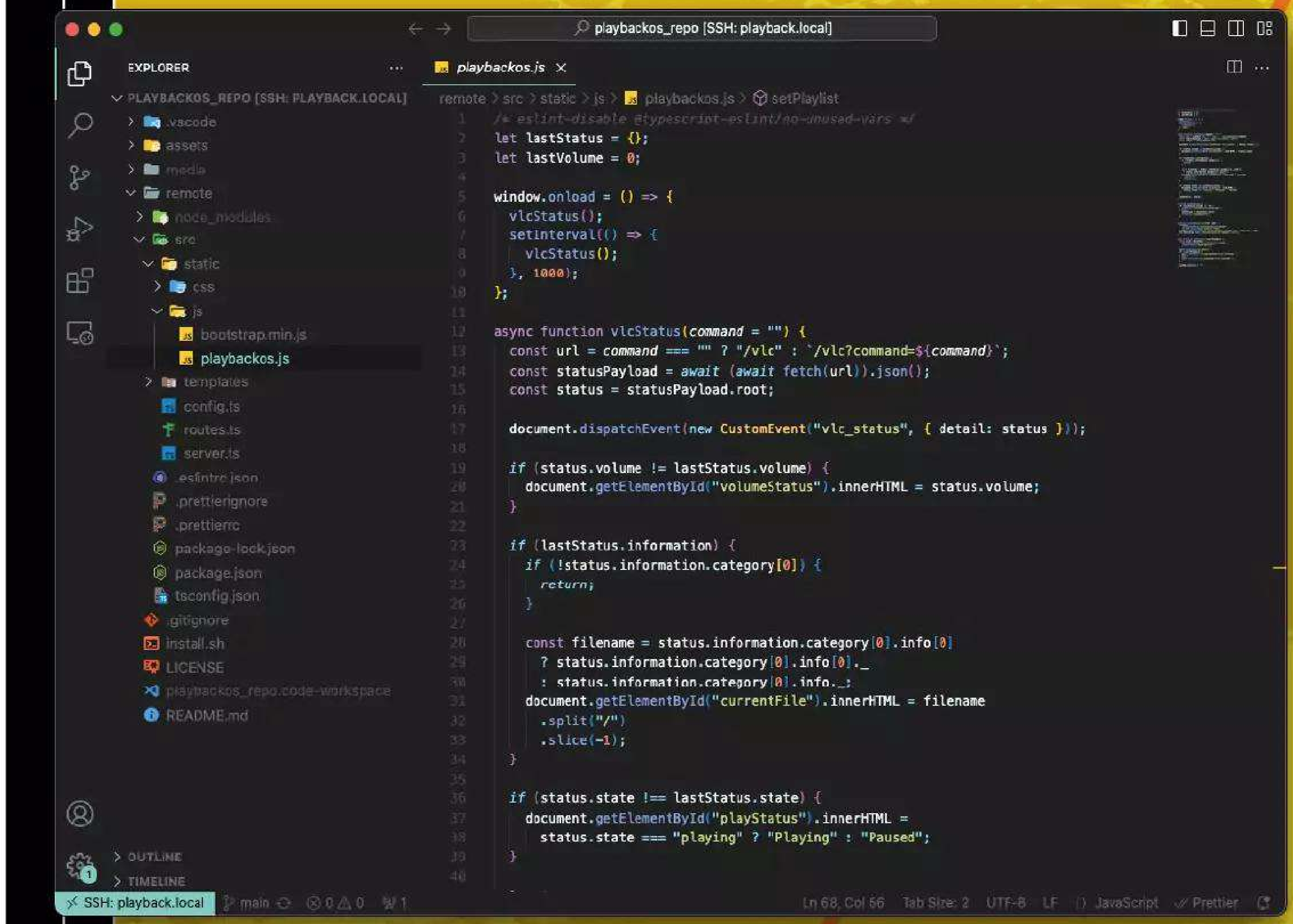
Install Visual Studio Code

Without a doubt, the most popular IDE in the world is Visual Studio Code (also known as 'VS Code'). Although not strictly a true IDE, it is a text editor with so much functionality that it blurs the line between the two concepts. Once you have graduated from Scratch and Thonny, then VS Code is your next destination.

As well as colour-coding of syntax to help you read your code, VS Code comes with an impressive range of keyboard shortcuts that make a developer's life so much simpler. Where it really shines though, is the Extensions Marketplace, which allows you to customise your experience to a surprising degree. As anyone can submit an extension to the marketplace, if you want VS Code to do something, chances are it's already available. From colour-coding CSV files to deploying web services or automatically formatting your code to keep it tidy, VS Code is a Swiss army knife for any developer.

Did we mention it's free? So are most of the extensions too. You can install VS Code on Raspberry Pi OS (Desktop) from the Recommend Applications app to get started. One of our favourite extensions is SSH, which allows you to develop and run code on a Raspberry Pi from any other machine on the network, so you can code for your project on any computer as if using Raspberry Pi itself.

> " Without a doubt, the most popular IDE in the world is Visual Studio Code "



```
/* eslint-disable @typescript-eslint/no-unused-vars */
let lastStatus = {};
let lastVolume = 0;

window.onload = () => {
  vlcStatus();
  setInterval(() => {
    vlcStatus();
  }, 1000);
};

async function vlcStatus(command = "") {
  const url = command === "" ? "/vlc" : `/vlc?command=${command}`;
  const statusPayload = await (await fetch(url)).json();
  const status = statusPayload.root;

  document.dispatchEvent(new CustomEvent("vlc_status", { detail: status }));

  if (status.volume != lastStatus.volume) {
    document.getElementById("volumeStatus").innerHTML = status.volume;
  }

  if (lastStatus.information) {
    if (!status.information.category[0]) {
      return;
    }

    const filename = status.information.category[0].info[0]
      ? status.information.category[0].info[0]._
      : status.information.category[0].info._;
    document.getElementById("currentFile").innerHTML = filename
      .split("/")
      .slice(-1);
  }

  if (status.state !== lastStatus.state) {
    document.getElementById("playStatus").innerHTML =
      status.state === "playing" ? "Playing" : "Paused";
  }
```

Learn to debug

Bugs happen. It's one of the first lessons you learn. Professional developers see bugs as part of the process; finding out what doesn't work. So don't be discouraged when your code doesn't run first time.

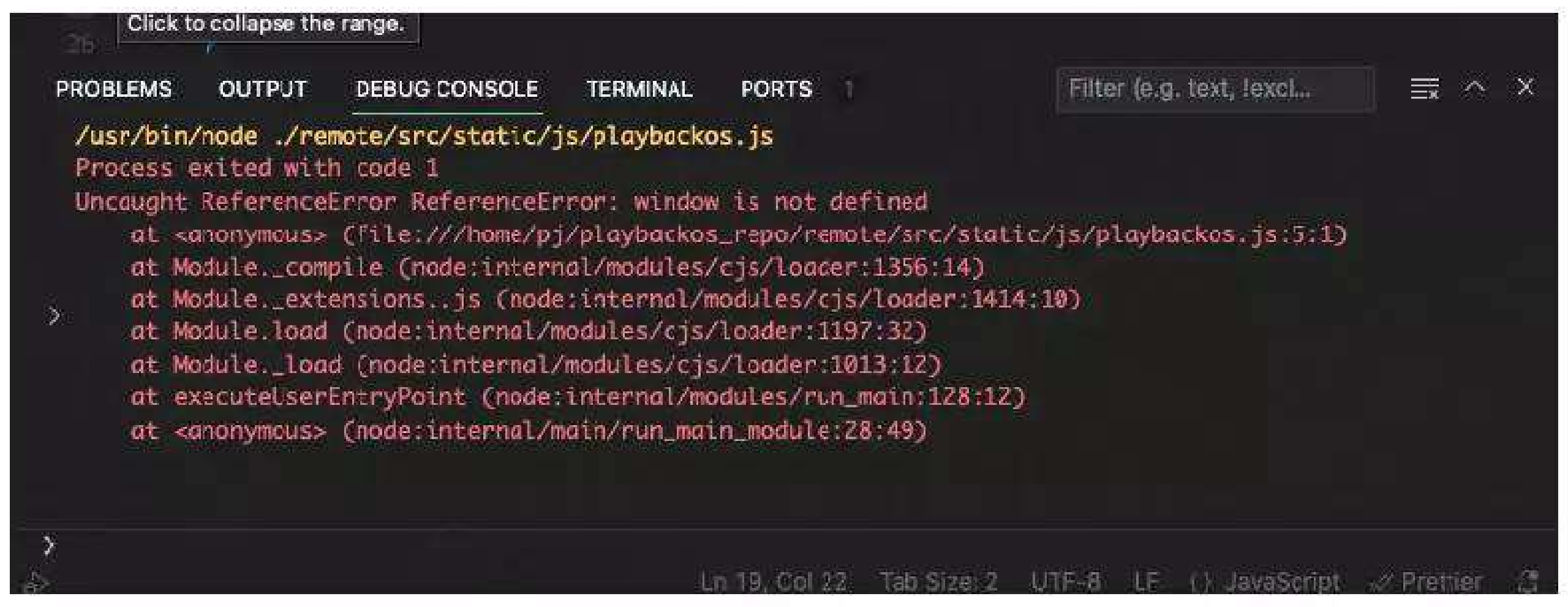
A bug can take on many forms, whether it's a simple typo stopping the code from running or a miscalculation which means the code runs, but produces the wrong result.

Removing bugs from your code (debugging) is a core part of any coder's experience. The struggle comes in two parts: Finding the bit of code that is misbehaving, and then fixing it. Each has its own challenges.

If your code is not behaving as expected, see if the editor or IDE you are using has a debugger, the ability to run the code line-by-line or halt it at a certain point so you can inspect it. This stepping-though can lead to an 'ah ha!' moment when the code heads off in a different direction than expected.

A tried-and-tested technique is monitoring the output of code by adding logging statements. For example, in a Python command-line app, adding `print("HERE")` or `print(variable)` can help you track what is going on. It's a rudimentary technique, but millions of progression coders use it every day to find problems.

When you've found your bug, it might not be obvious how to fix it. Sometimes it can be, such as "Oh, I used a + instead of a - to calculate the total". If you find yourself stuck, you may need to refactor: the process of scrapping a section of code and rewriting it to provide a better solution.



```
/usr/bin/node ./remote/src/static/js/playbackos.js
Process exited with code 1
Uncaught ReferenceError ReferenceError: window is not defined
    at <anonymous> (file:///home/pj/playbackos_repo/remote/src/static/js/playbackos.js:5:1)
    at Module._compile (node:internal/modules/cjs/loader:1356:14)
    at Module._extensions..js (node:internal/modules/cjs/loader:1414:10)
    at Module.load (node:internal/modules/cjs/loader:1197:32)
    at Module._load (node:internal/modules/cjs/loader:1013:12)
    at executeUserEntryPoint (node:internal/modules/run_main:128:12)
    at <anonymous> (node:internal/main/run_main_module:28:49)
```

▲ Example error output from VS Code's debugging console

Jargon buster

Some common terms you'll come across

Variable

A named token representing text or a numeric value. Used to store information for later use.

```
pepperoniSlicesPerPizza = 10
```

Function

Code grouped together to perform a function that can be used multiple times without repeating yourself.

```
def makePizza(type):
        Ingredients = pizzas[type].toppings;
```

Conditionals, Ifs

A piece of code that makes a decision

```
If (type.cheese):
        addCheese()
```

Loops

An instruction that repeats a block of code until a certain condition is met

```
for (topping in toppings):
        addIngredient(topping)
```

Class and Object

A collection of variables and functions structured in a logical group that combines both variables and functions.

Array, Dict, List

A list of related variables

```
pizzas = [pepperoni,  hawaiian, margherita]
```

Modules, Extensions

The ability to add new or enhanced functionality to a programming language using existing public code

```
from pizzalib import makePizza
```

Compiling

Converting code text into executable binary code for faster execution

Starting tips

- **Be fearless**
  It doesn't matter if you break something, just start again. You can't hurt the computer.

- **Start small**
  Don't try and write an Excel competitor on your first outing. Find a small problem and see if you can solve it.

- **Keep it simple**
  Never write more code than you need to. In coding this is known 'over-engineering' or YAGNI (you ain't gonna need it).

- **Separate concerns**
  This means that any block of code should do one job and one job only. To do otherwise is to create 'spaghetti' code.

- **Search for it**
  Some professional developers joke that their real skill is using search engines. There is a wealth of information out there, and if you've got stuck, the chances are someone else has solved the same problem.

- **Don't repeat yourself (DRY)**
  If you find you're repeating the same bit of code multiple times, refactor it into a function and call that instead. Doing this means you've only got one place to fix bugs.

- **Name things sensibly**
  Another common tenet of coding is 'Naming things is hard'. Truth is, it isn't, but people like shortcuts. Never call a variable 'x' or 'y', tell a story instead: 'pizzaOrderGrandTotal' tells you everything you need to know.

Learning resources

There are many online 'boot camps' for learning to code and they cover all ranges of experience. Here are some of our favourites:

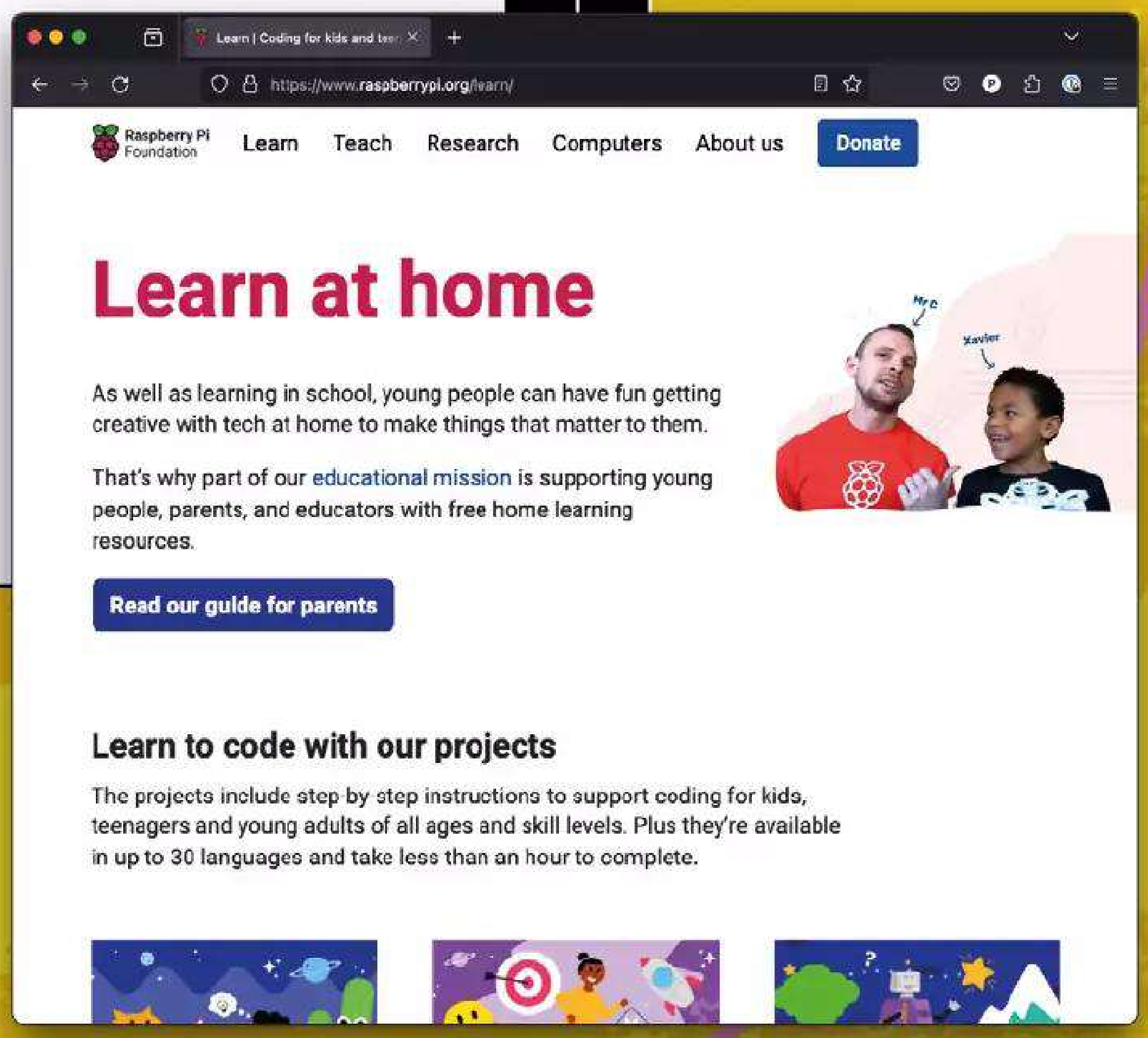
- **FreeCodeCamp**
  A huge and, as the name suggests, free collection of courses suitable for the absolute beginner
  freecodecamp.org

- **Raspberry Pi resources**
  Our very own Raspberry Pi Foundation is a suburb resource, especially for Scratch and Python
  raspberrypi.org/learn

- **Learn Python**
  A complete course in Python where you code in the browser!
  learnpython.org

- **Nodeschool**
  Learn Node for free with this collection of self-paced workshops
  nodeschool.io

- **Go Learn**
  The official resource of the Go project with beginner tutorials available
  go.dev/learn


Learn | Coding for kids and tee
https://www.raspberrypi.org/learn/
Raspberry Pi Foundation
Learn
Teach
Research
Computers
About us
Donate
Mr C
Xavier
Read our guide for parents

Learn at home

As well as learning in school, young people can have fun getting creative with tech at home to make things that matter to them.

That's why part of our educational mission is supporting young people, parents, and educators with free home learning resources.

Learn to code with our projects

The projects include step by step instructions to support coding for kids, teenagers and young adults of all ages and skill levels. Plus they're available in up to 30 languages and take less than an hour to complete.

Code daily

Like learning any new skill, daily practice yields benefits and increases your 'muscle memory' for coding. Try writing a small app every day. If you need inspiration, there are many coding challenges out there such as Free Code Camp **freecodecamp.org** or the trickier Advent Of Code: **adventofcode.com**.

It's important not to be intimidated by the potential complexity of learning a programming language. Set yourself a small goal and build from there. It's no coincidence that the first thing we generally do is print 'Hello world!' to the screen. You just wrote your first program! Coding can, and should, be fun and rewarding.

Part 01

Learn Python: Make a To-Do List

Keep track of your tasks with a simple to-do list. This program explains how the basics of Python programming work

MAKER

Lucy Hattersley

Lucy is editor of The MagPi and she loves a good to-do list.

magpi.cc

You'll Need

- Raspberry Pi
- Raspberry Pi OS
- Thonny IDE

Python is an incredibly versatile programming language. It's great for beginners because it is easy to read and understand, yet Python is also powerful enough for data scientists and AI experts.

While there are many other programming languages out there, and each has its own merits depending on what you're trying to do, we think Python is the one to start with.

In this tutorial, we're going to show you how to make a to-do list application in Python. It won't be the world's best to-do list. That's not the point. You've made it yourself and will start to understand the fundamental building blocks of programming languages.

Here are some of the concepts we'll be looking at

- **Variables:** these are used to store items (in this case our to-dos)
- **Functions:** these are used to quickly repeat code
- **Conditions:** Used to direct code in directions to choose different functions
- **Loops:** Used to repeat code

We will bring this all together in our **todo.py** code. Let's get started.

Start by opening Thonny IDE (Raspberry Pi menu > Programming > Thonny). We can write programs in any text editor, but Thonny is very friendly for beginners and lets us take a look at elements of it in closer detail.

> "Python is powerful enough for data scientists and AI experts"

It's tradition to start any new programming language by displaying the text line "Hello, World!". Enter the following code into the Code View in the top half of Thonny

```
print("Hello, World!")
```

We should save this. Click Save and give it the name **hello_world.py**. The '.py' extension indicates this is a Python program and by convention, all Python file names are lowercase with words separated with underscores.

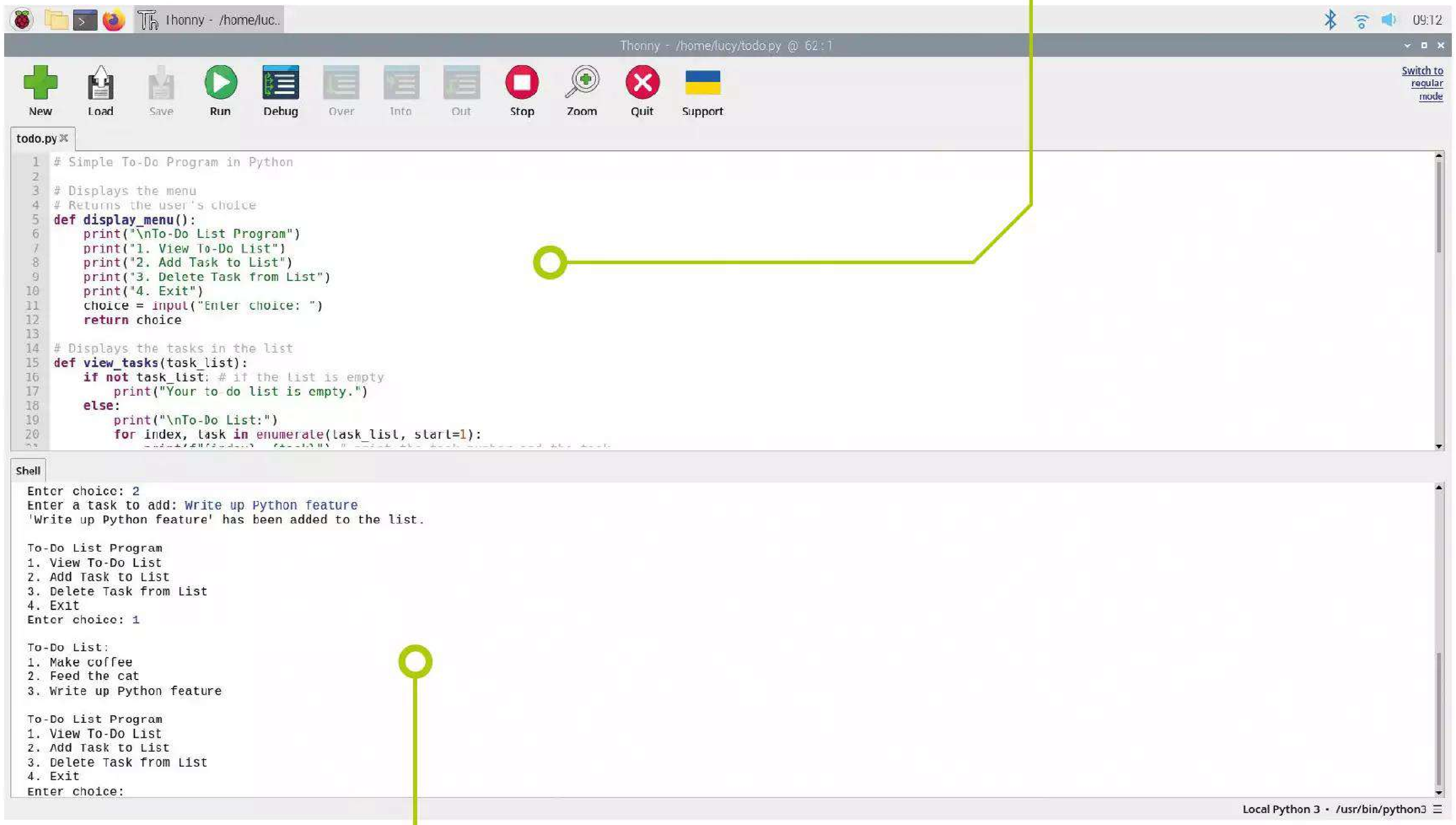
Click Run and the words "Hello, World!" will appear in the shell below. Congratulations on running your first program!

It's all variable

A key concept to learn in programming is variables (sometimes called 'vars' for short). These are storage containers in your program used to save (and reuse) items. These can be different types: typically numbers, words and Boolean statements. Numbers are known as 'integers' or 'ints' if they are round numbers, or 'floats' if they have a decimal place. And words are called 'strings' in computer parlance. Boolean operators are 'True/False' statements used in decision-making.

Create a new program called **vars.py** and enter the following code:

The Code Editor is where you enter code. Press Run in the Tools Menu to start your program



```
# Simple To-Do Program in Python

# Displays the menu
# Returns the user's choice
def display_menu():
    print("\nTo-Do List Program")
    print("1. View To-Do List")
    print("2. Add Task to List")
    print("3. Delete Task from List")
    print("4. Exit")
    choice = input("Enter choice: ")
    return choice

# Displays the tasks in the list
def view_tasks(task_list):
    if not task_list: # if the list is empty
        print("Your to do list is empty.")
    else:
        print("\nTo-Do List:")
        for index, task in enumerate(task_list, start=1):
```

```
Enter choice: 2
Enter a task to add: Write up Python feature
'Write up Python feature' has been added to the list.

To-Do List Program
1. View To-Do List
2. Add Task to List
3. Delete Task from List
4. Exit
Enter choice: 1

To-Do List:
1. Make coffee
2. Feed the cat
3. Write up Python feature

To-Do List Program
1. View To-Do List
2. Add Task to List
3. Delete Task from List
4. Exit
Enter choice:
```

The Shell in the bottom of the interface displays the output from your code. You also type commands here to interact with the program

```
name = "John" # String
age = 30 # Integer
height = 5.9 # Float
student = True # Boolean
print(f"Name: {name}, Age: {age}, Height:
{height}m, Student: {student}")
```

This program has four different types of variables (indicated in the comments). Notice that our print statement here is more complex than Hello World. It starts with `print(f` which indicates this is a 'formatted string literal' also known as an 'f-string'. The `f` itself isn't printed, but Python now knows that the words inside curly brackets (such as `{name}` refers to a variable and inserts its value into the output (so `{name}` becomes 'john').

Syntax and indents

When writing out a program it's important to understand the syntax. This is the structure of the code. Be sure to enter code exactly as you see it, and we find it helps to read our programs from the last line to the first line to check for any errors. If you get any errors read through carefully and fix any problems.

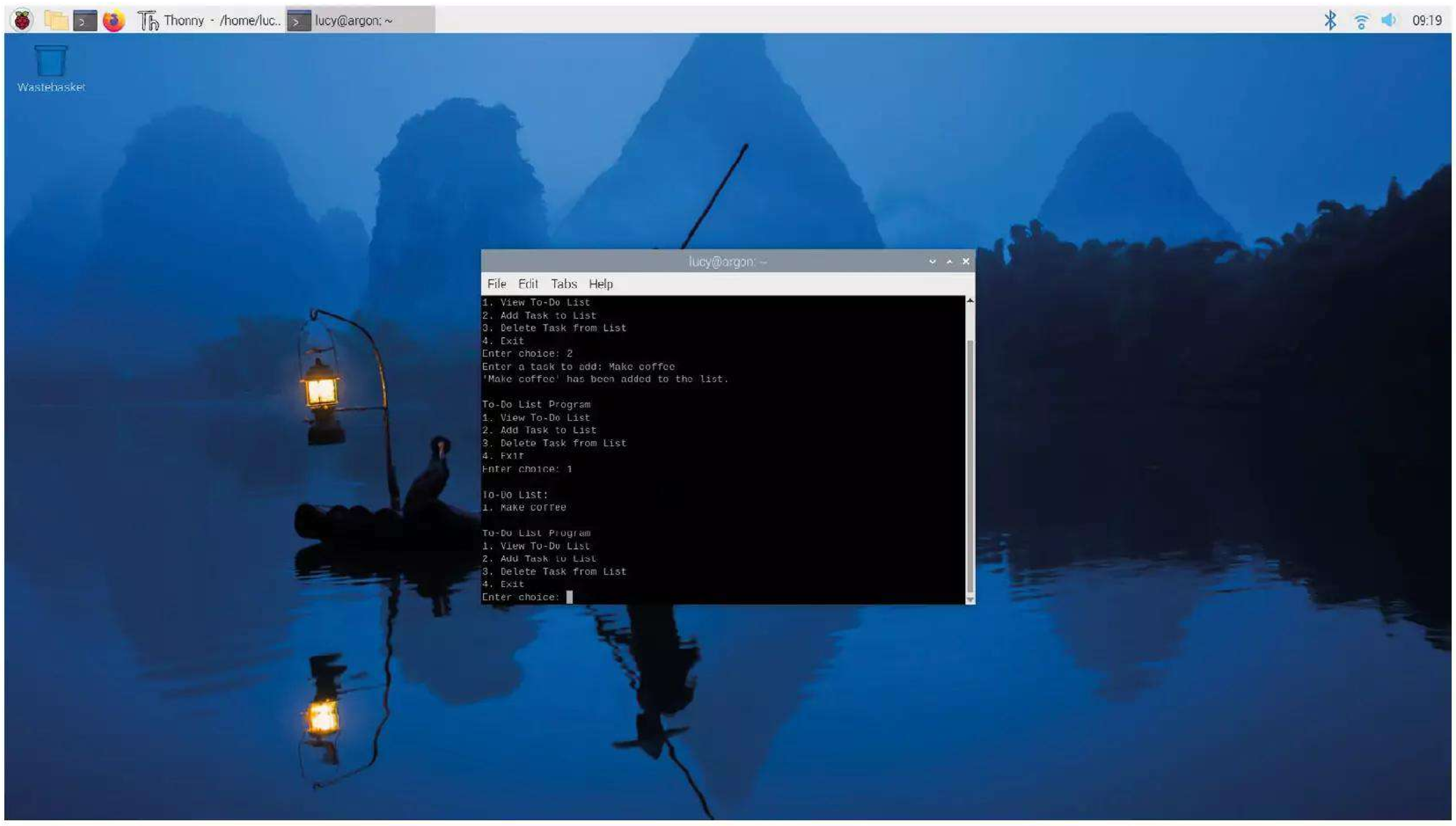
The next thing we're going to look at is called an indentation. Python programs are often indented with spaces at the start of some lines. These aren't just to look pretty, they indicate a relationship between a line of code and the following lines.

In Python, it is always blocks of four spaces. Always press the **SPACE** key four times, never use **TAB**. If you have more than one layer of

Top Tip

Be exact

When typing out code be sure to type the code exactly as you see it. Use the same spaces and capital/lowercase letters.



```
1. View To-Do List
2. Add Task to List
3. Delete Task from List
4. Exit
Enter choice: 2
Enter a task to add: Make coffee
'Make coffee' has been added to the list.

To-Do List Program
1. View To-Do List
2. Add Task to List
3. Delete Task from List
4. Exit
Enter choice: 1

To-Do List:
1. Make coffee

To-Do List Program
1. View To-Do List
2. Add Task to List
3. Delete Task from List
4. Exit
Enter choice: 
```

▶ Python programs can also be run directly from a Terminal window

> “When writing out a program it's important to understand the syntax”

indentation you press the **SPACE** key 8 times, 12 times and so on (it's generally considered pretty bad to have too much indentation though, as it becomes hard to understand).

Indentation is used to indicate conditional statements, loops, and to define functions. Let's start with a conditional statement.

It's a condition

A conditional statement is often called an if/else statement (there is also a third: else-if component). The basic idea is this. If the first line is True, then run the code below that is indented by four lines:

```
if condition:
    # code to execute if condition is true
```

Typically if/else statements will has have alternative options. So `if condition1` is True then execute the line indented, `else` run the code indented below instead.

```
if condition1:
    # code to execute if condition1 is true
else:
    # code to execute if condition1 is false
```

Let's put this into practice:

```
name = "John" # String
student = True # Boolean

if (student == True):
    print(f"Name: {name} is a student!")
else:
    print(f"Name: {name} is not a student!")
```

Run this code and you will see “Name: John is a student!” The important part here is that the code below the ‘else’ statement is not run. Indentation is how decisions in Python programs are made. Try changing True to False on line 2 and running the program again.

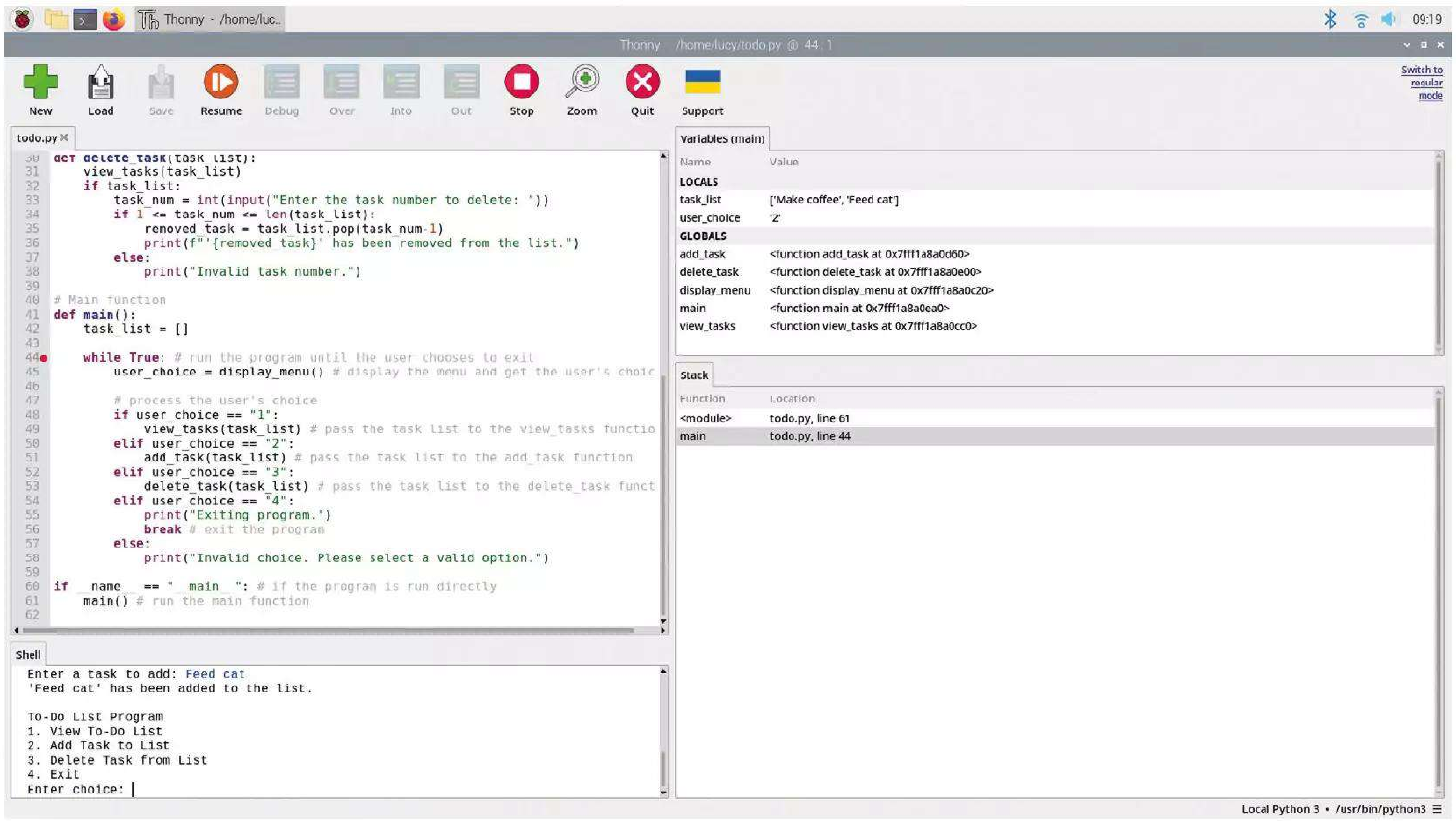
Note the `==` symbol in (`student == True`). This is the ‘equality operator’ which checks if the expression evaluates to true.

It's different to the `=` symbol in `name = "John"` and `student = True`. This single = sign is the ‘assignment operator’ which assigns the value ‘If john’ and True to the variables.

Top Tip

PEP Style Guide

Read and bookmark the Python style guide **magpi.cc/pep8**



```
def delete_task(task_list):
    view_tasks(task_list)
    if task_list:
        task_num = int(input("Enter the task number to delete: "))
        if 1 <= task_num <= len(task_list):
            removed_task = task_list.pop(task_num-1)
            print(f"'{removed_task}' has been removed from the list.")
        else:
            print("Invalid task number.")

# Main function
def main():
    task_list = []

    while True: # run the program until the user chooses to exit
        user_choice = display_menu() # display the menu and get the user's choic

        # process the user's choice
        if user_choice == "1":
            view_tasks(task_list) # pass the task list to the view_tasks functio
        elif user_choice == "2":
            add_task(task_list) # pass the task list to the add_task function
        elif user_choice == "3":
            delete_task(task_list) # pass the task list to the delete_task funct
        elif user_choice == "4":
            print("Exiting program.")
            break # exit the program
        else:
            print("Invalid choice. Please select a valid option.")

if __name__ == "__main__": # if the program is run directly
    main() # run the main function
```

```
Enter a task to add: Feed cat
'Feed cat' has been added to the list.

To-Do List Program
1. View To-Do List
2. Add Task to List
3. Delete Task from List
4. Exit
Enter choice: 
```

Variables (main)

Name	Value
LOCALS	
task_list	['Make coffee', 'Feed cat']
user_choice	'2'
GLOBALS	
add_task	<function add_task at 0x7fff1a8a0d60>
delete_task	<function delete_task at 0x7fff1a8a0e00>
display_menu	<function display_menu at 0x7fff1a8a0c20>
main	<function main at 0x7fff1a8a0ea0>
view_tasks	<function view_tasks at 0x7fff1a8a0cc0>

Stack

Function	Location
<module>	todo.py, line 61
main	todo.py, line 44

Using Debug instead of Run enables you to investigate the code and look at the variables

> “Indentation is used to indicate conditional statements, loops, and define functions”

It’s easy to get the two mixed up, so use them carefully.

Let’s create a more complicated decision tree using `if`, `elif` and `else` statements and a comparison operator called ‘less-than’ indicated by a `<` symbol.

```
name = "John" # String
age = 23 # Integer

if (age < 16):
    print(f"Name: {name} is under 16!")
elif (age < 25):
    print(f"Name: {name} is under 25!")
else:
    print(f"Name: {name} is over 25!")
```

Here each line is checking if the age is less than a value. And evaluating to True if the integer in the age variable is less than the integer in the conditional statement.

Play around with the age variable to run the different options.

Loop the loop

Loops are similar in structure to if/else statements and are also indented by four lines. In a loop, however, the indented code is designed to run multiple times based on a condition. There are different types of loops: our To-Do program features a ‘while’ loop which loops indefinitely until a condition is met:

```
while condition:
    # Block of code to execute
```

Let’s create a simple countdown program that counts down from 10 to 1 then says “Blast off!”.

```
counter = 10 # initialize the counter
while counter > 0: # loop as long as the
counter is greater than 0
    print(counter)
    counter -= 1 # decrement the counter
print("Blast off!")
```

Our To-Do program uses a `while True` condition with a `break` statement. This runs the program until we choose to exit. When using `while True` be careful to allow your program a means to exit, otherwise, you are creating an infinite loop, which sounds cool but is generally considered bad form.

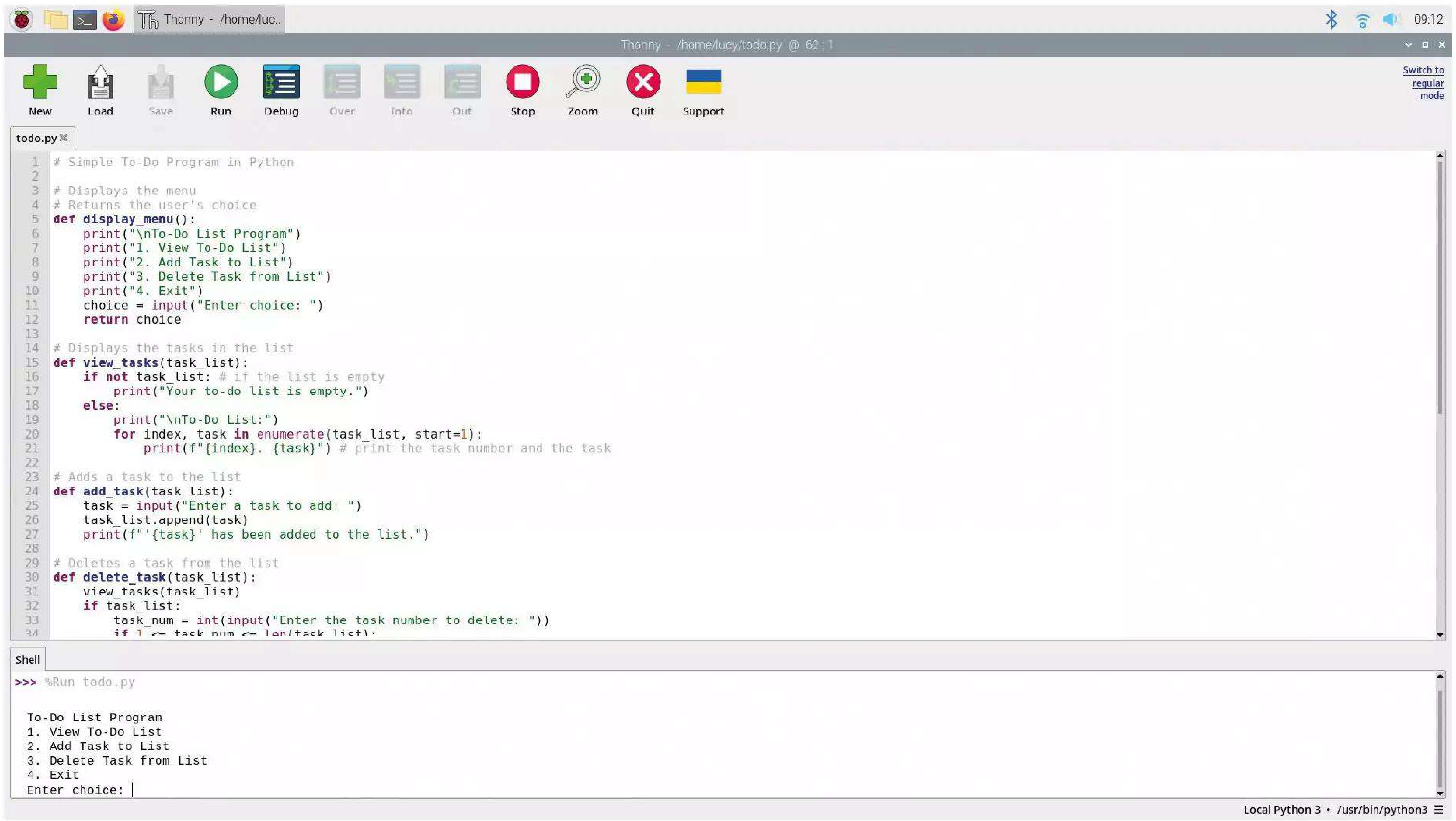
Executive function

The third type of indented code in Python is called a Function. This is a block of reusable code that can be run again and again simply by writing the function name. Think of it like copy and paste for code.

Top Tip

Spaces, not tabs

Always use four spaces to indent your code instead of pressing **TAB**. There are arguments about this, but beginners should always start with spaces.



```
# Simple To-Do Program in Python

# Displays the menu
# Returns the user's choice
def display_menu():
    print("\nTo-Do List Program")
    print("1. View To-Do List")
    print("2. Add Task to List")
    print("3. Delete Task from List")
    print("4. Exit")
    choice = input("Enter choice: ")
    return choice

# Displays the tasks in the list
def view_tasks(task_list):
    if not task_list: # if the list is empty
        print("Your to-do list is empty.")
    else:
        print("\nTo-Do List:")
        for index, task in enumerate(task_list, start=1):
            print(f"{index}. {task}") # print the task number and the task

# Adds a task to the list
def add_task(task_list):
    task = input("Enter a task to add: ")
    task_list.append(task)
    print(f"'{task}' has been added to the list.")

# Deletes a task from the list
def delete_task(task_list):
    view_tasks(task_list)
    if task_list:
        task_num = int(input("Enter the task number to delete: "))
        if 1 <= task_num <= len(task_list):
```

```
>>> %Run todo.py

 To-Do List Program
 1. View To-Do List
 2. Add Task to List
 3. Delete Task from List
 4. Exit
 Enter choice: 
```

Thonny is an easy-to-use IDE (integrated development environment) installed by default in Raspberry Pi OS

A function is defined using `def`:

```
def function_name(parameters):
    # Block of code
```

You call the function using the `function_name` command and add any optional items into the brackets. You've already seen a function built into Python: `print()`.

When you first entered `print("Hello, World!")` you used `print()` as a function command and passed "Hello, World" as an argument into the parameters. The print function then outputs "Hello, World" as a string.

The `def` command lets you create your bespoke functions. Let's create one that greets the person passed into the parameter:

```
def greet(name): # name is a parameter
    print(f"Hello, {name}!")
message = greet("Alice") # Alice is an argument
message = greet("John") # John is an argument
```

Run this code and you will see "Hello, Alice!" and "Hello, John!". It's running the same `print()` command each time but with a different argument.

Wrapping up

There's a whole lot more to Python and other coding languages but fundamentally it all breaks down into Functions, Loops and Conditions. Re-running the code again and again, slightly differently, and making different decisions based on what is happening.

We hope you enjoy the journey. We love programming and it helps us to think, and problem-solve. These are the foundational building blocks of programming and now you can identify them Python programming will start to open up.

Enter each line of **todo.py** code carefully and think about what each line does for the program. Run it and if you get an error read your code carefully and adjust the incorrect line. When you have it up and running play around with it.

Comments

Comments in Python are used to explain what the code is supposed to do, making it easier for someone to understand the code at a glance. The Python interpreter does not execute comments. They're used solely for documentation purposes.

It is good practice to add comments to explain your code. Even if you are just helping to explain it to yourself.

Comments in Python are indicated with a # symbol and the rest of the line is ignored:

```
# This is a single-line comment
print("Hello, World!")  # This is also a
comment, following code on the same line
```

Don't skimp on comments. They are a sign of good code and it's best to get into the habit early.
magpi.cc/pepcomments

todo.py

Language: **Python**

DOWNLOAD THE FULL CODE:

magpi.cc/github

```python
# Simple To-Do Program in Python

# Displays the menu
# Returns the user’s choice
def display_menu():
    print("\nTo-Do List Program")
    print("1. View To-Do List")
    print("2. Add Task to List")
    print("3. Delete Task from List")
    print("4. Exit")
    choice = input("Enter choice: ")
    return choice

# Displays the tasks in the list
def view_tasks(task_list):
    if not task_list: # if the list is empty
        print("Your to-do list is empty.")
    else:
        print("\nTo-Do List:")
        for index, task in enumerate(task_list, start=1):
            print(f"{index}. {task}") # print the task number and the task

# Adds a task to the list
def add_task(task_list):
    task = input("Enter a task to add: ")
    task_list.append(task)
    print(f"’{task}’ has been added to the list.")

# Deletes a task from the list
def delete_task(task_list):
    view_tasks(task_list)
    if task_list:
        task_num = int(input("Enter the task number to delete: "))
        if 1 <= task_num <= len(task_list):
            removed_task = task_list.pop(task_num-1)
            print(f"’{removed_task}’ has been removed from the list.")
        else:
            print("Invalid task number.")

# Main function
def main():
    task_list = []

    while True: # run the program until the user chooses to exit
        user_choice = display_menu() # display the menu and get the user’s choice

        # process the user’s choice
        if user_choice == "1":
            view_tasks(task_list) # pass the task list to the view_tasks function
        elif user_choice == "2":
            add_task(task_list) # pass the task list to the add_task function
        elif user_choice == "3":
            delete_task(task_list) # pass the task list to the delete_task function
        elif user_choice == "4":
            print("Exiting program.")
            break # exit the program
        else:
            print("Invalid choice. Please select a valid option.")

if __name__ == "__main__": # if the program is run directly
    main() # run the main function
```

Part 02

Getting started with your Raspberry Pi

Discover the essential items you'll need for your Raspberry Pi, and learn how to connect them all to get it set up and working

MAKER

Gareth Halfacree

Gareth is a freelance technology journalist, writer, and former system administrator in the education sector with a passion for open-source software and hardware.

freelance.halfacree.co.uk

Raspberry Pi is designed to be as quick and easy to set up and use as possible, but – like any computer – it relies on various external components called peripherals. While it's easy to take a look at the bare circuit board of Raspberry Pi – which looks significantly different to the encased, closed-off computers you may be used to – and worry that things are about to get complicated, that's not the case. You can be up and running with your Raspberry Pi in well under ten minutes if you follow the steps in this guide.

If you purchased a Raspberry Pi Desktop Kit or a Raspberry Pi 400, you'll already have almost everything you need to get started. All you need to provide is a computer monitor or a TV with a HDMI connection – the same type of connector used by set-top boxes, Blu-ray players, and game consoles – so you can see what your Raspberry Pi is doing.

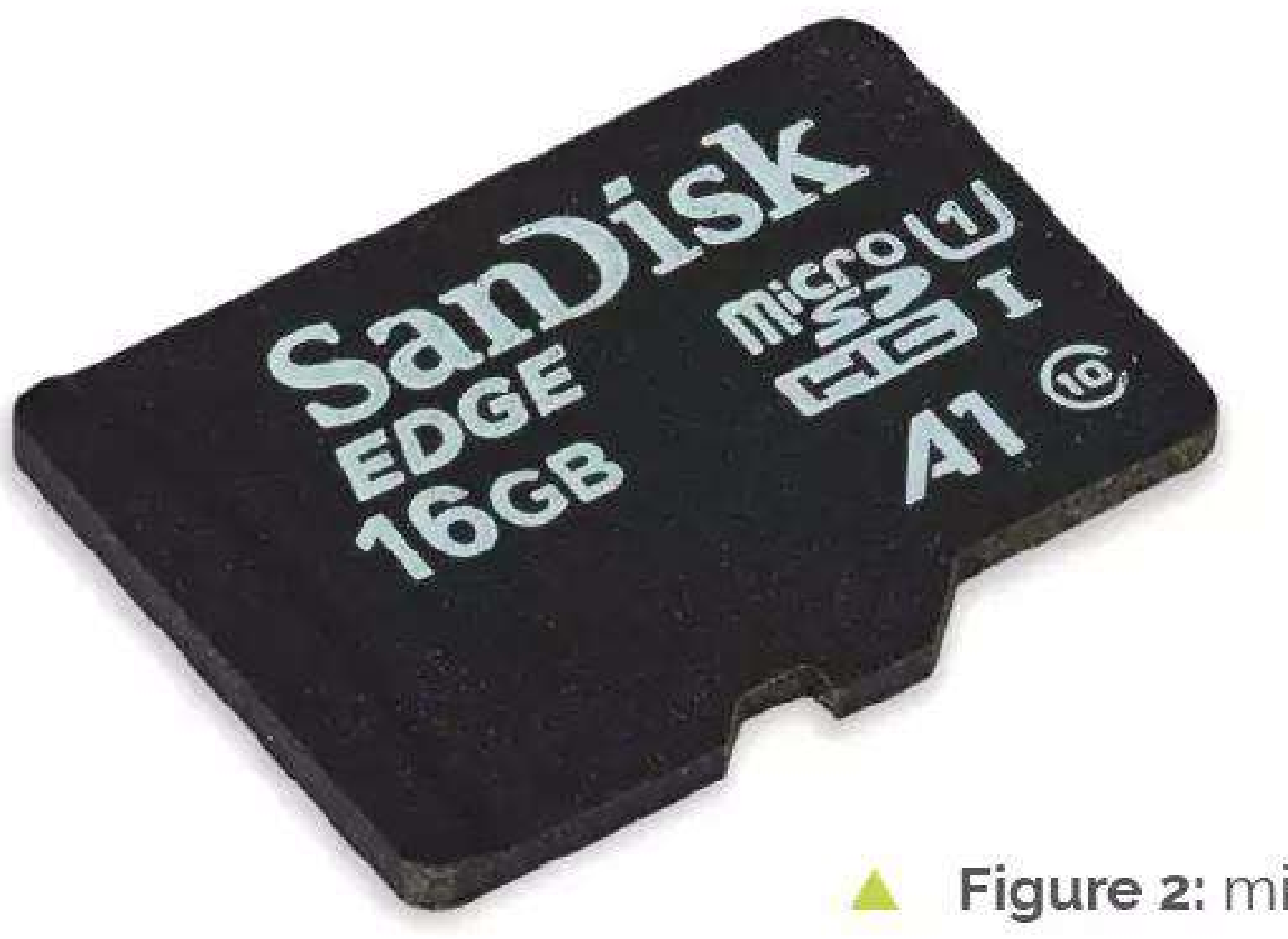
If you picked up your Raspberry Pi without accessories, then you'll also need:

1. USB power supply. A 5V power supply rated at 5 amps (5A) and with a USB-C connector for Raspberry Pi 5, a 5V power supply rated at 3 amps (3A) and with a USB-C connector for Raspberry Pi 4 Model B or Raspberry Pi 400, or a 5V power supply rated at 2.5 amps (2.5A) and with a micro-USB connector for Raspberry Pi Zero 2 W. The Official Raspberry Pi Power Supplies are recommended (**Figure 1**), as they are designed to cope with the quickly switching power demands of Raspberry Pi. Third-party power supplies may not be able to negotiate current, and may cause power issues with your Raspberry Pi.
2. microSD card – The microSD card (**Figure 2**) acts as your Raspberry Pi's permanent storage. All the files you create and all the software you install, along with the operating system itself, are stored on the card. An 8GB card is enough to get you started, though a 16GB one offers more room to grow. The Raspberry Pi Desktop Kit includes a microSD card with Raspberry Pi OS pre-installed.
3. USB keyboard and mouse. The keyboard and mouse (**Figure 3**) allow you to control your Raspberry Pi. Almost any wired or wireless keyboard and mouse with a USB connector will work with Raspberry Pi, though some gaming-style keyboards with colourful lights may draw too much power to be used reliably. Raspberry Pi Zero 2 W needs a micro-USB OTG adapter, and if you want to plug in more than one USB device at a time you'll need a powered USB hub.

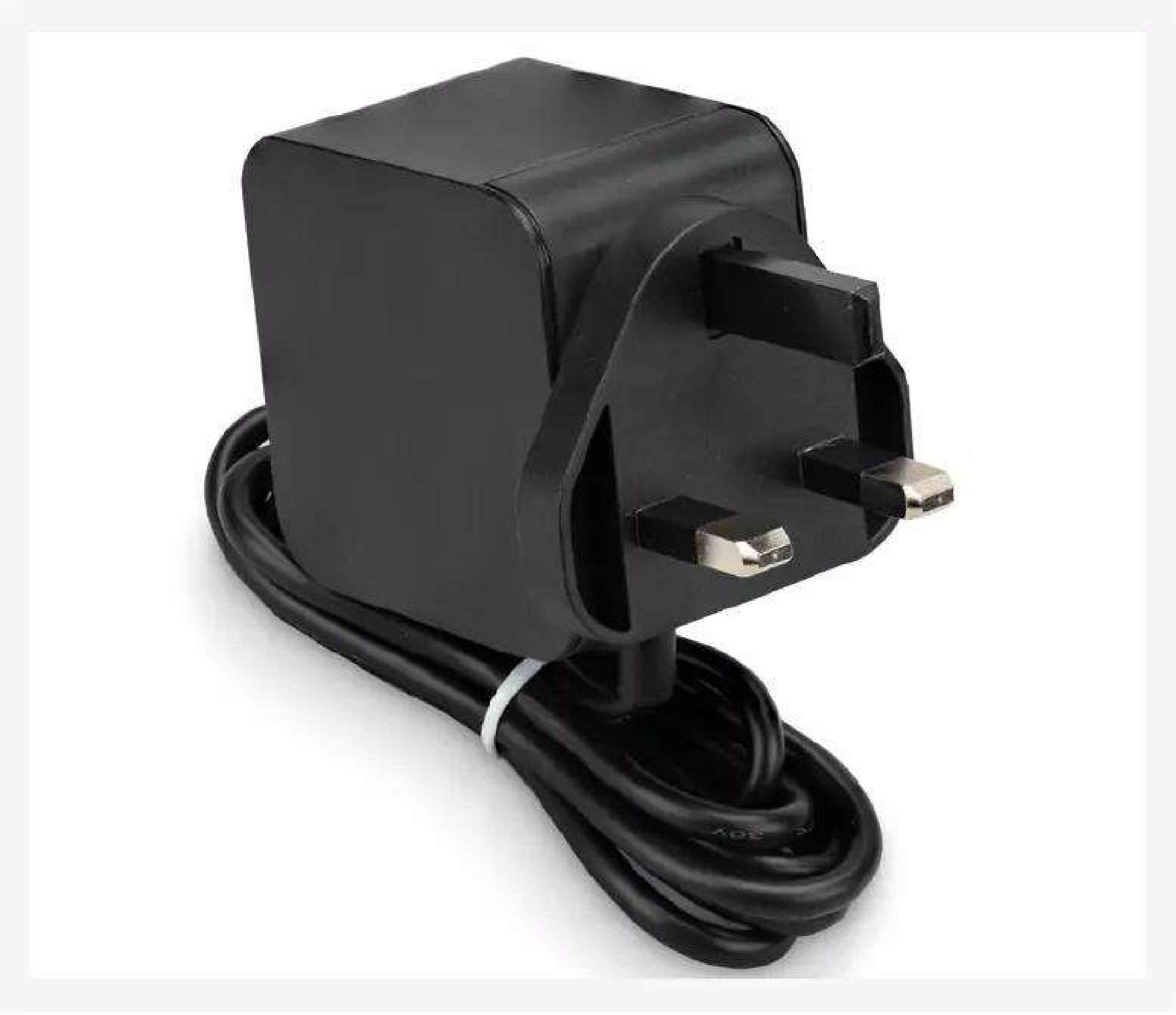
Warning! Power supply

Raspberry Pi 5 needs a 5V power supply capable of delivering 5A of current, and a suitable E-Marked USB C cable. If you connect a lower-current power supply, including the Official Raspberry Pi 4 Power Supply, Raspberry Pi 5's USB ports will be limited to low-power devices only.

magpi.cc/power



▲ **Figure 2:** microSD card



▲ **Figure:** 1 USB power supply

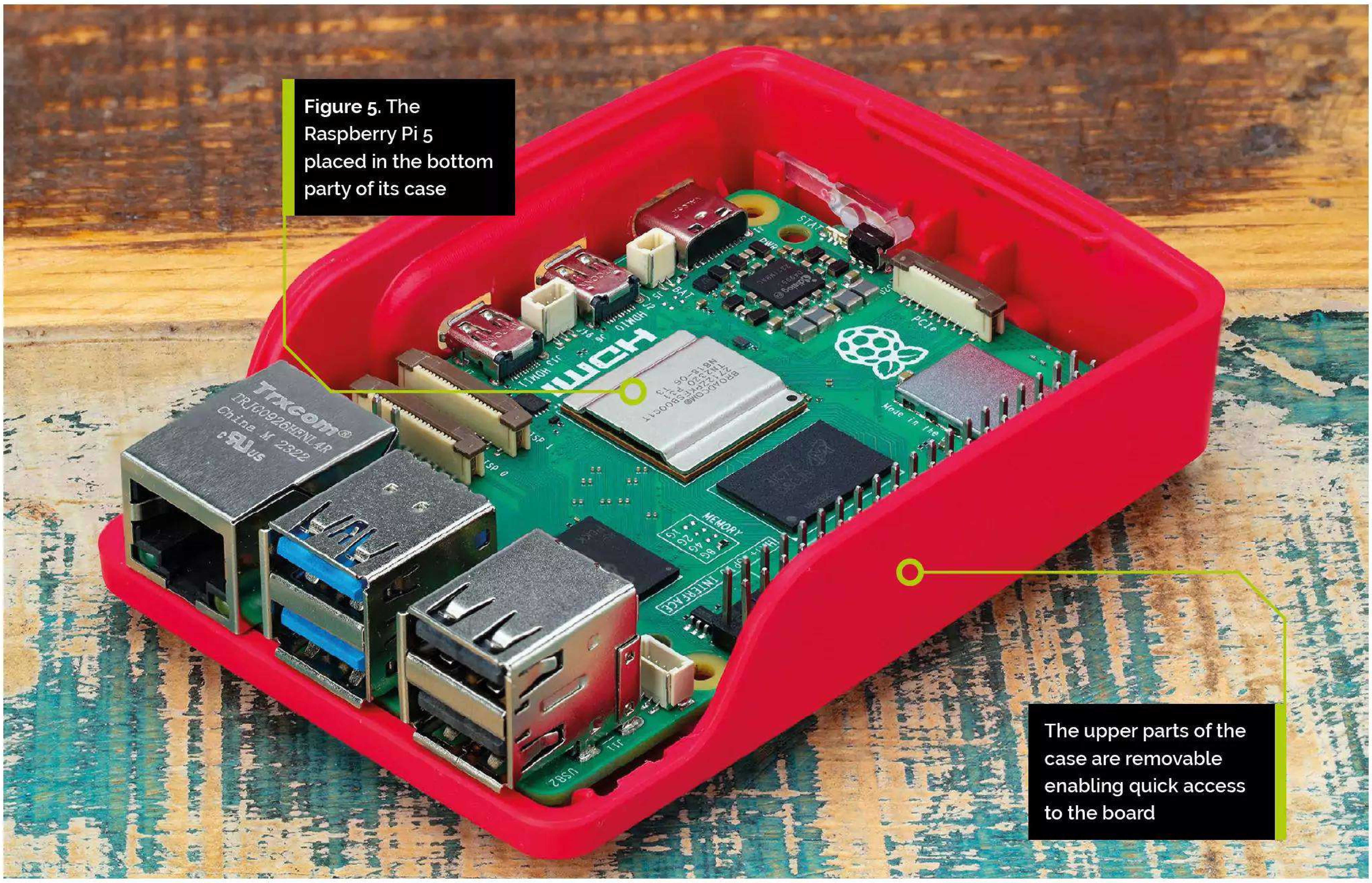


Figure 5. The Raspberry Pi 5 placed in the bottom party of its case

The upper parts of the case are removable enabling quick access to the board

4. HDMI cable. This carries sound and images from your Raspberry Pi to your TV or monitor. Raspberry Pi 4, Raspberry Pi 5, and Raspberry Pi 400 need a cable with a micro-HDMI connector at one end (**Figure 4**), while Raspberry Pi Zero 2 W needs a cable with a mini-HDMI connector; the other end should have a full-size HDMI connector for your display. You can also use a micro- or mini-HDMI to HDMI adapter along with a standard, full-size HDMI cable. If you're using a monitor without an HDMI socket, you can buy adapters to convert to DVI-D, DisplayPort, or VGA connectors.

▲ **Figure 3:** USB keyboard

▶ **Figure 4:** HDMI cable

Raspberry Pi is safe to use without a case, providing you don't place it on a metal surface, which could conduct electricity and cause a short-circuit. An optional case, however, can provide additional protection; the Desktop Kit includes the Official Raspberry Pi Case, while third-party cases are available from all good stockists.

If you want to use Raspberry Pi 4, Raspberry Pi 5, or Raspberry Pi 400 on a wired network, rather than a Wi-Fi network, you'll also need an Ethernet network cable. This should be connected at one end to your network's switch or router. If you're planning to use Raspberry Pi's built-in wireless radio, you won't need a cable; you will, however, need to know the name and key or passphrase for your wireless network.

Setting up the hardware

Begin by unpacking your Raspberry Pi from its box. Raspberry Pi is a robust piece of hardware, but that doesn't mean it's indestructible; try to get into the habit of holding the board by the edges, rather than on its flat sides, and be extra careful around the raised metal pins. If these pins are bent, at best it'll make using add-on boards and other extra hardware difficult and,

You'll Need

- Raspberry Pi
- Raspberry Pi OS

Top Tip

Raspberry Pi 400 setup

The following instructions are for setting up Raspberry Pi 5 or another bare-board member of the Raspberry Pi family. For Raspberry Pi 400, see Raspberry Pi 400 setup instructions.

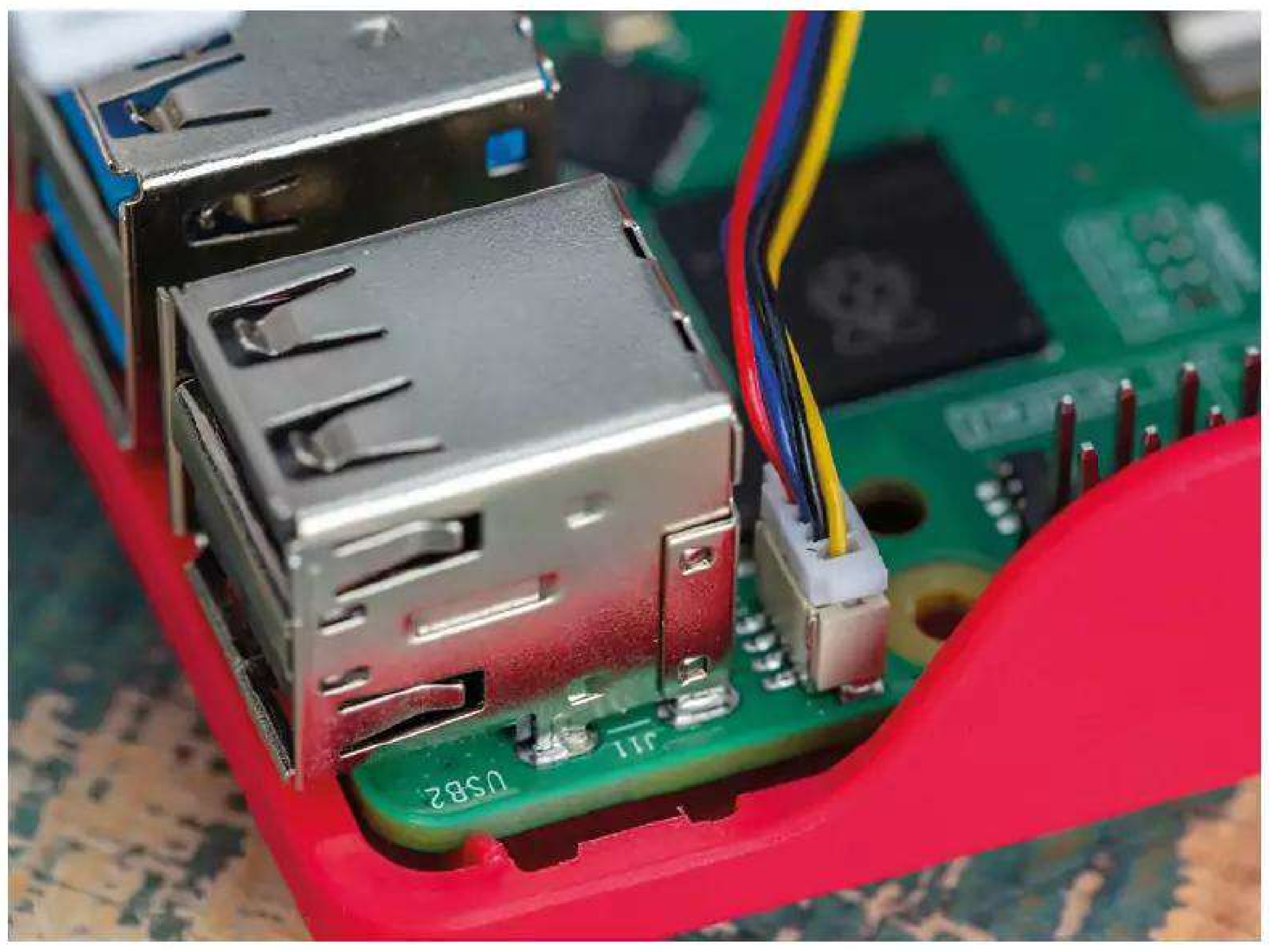


Figure 6: Plugging in the fan connector

at worst, might cause a short-circuit that will damage your Raspberry Pi.

If you haven't done so already, have a look *Get to know your Raspberry Pi* in The MagPi issue #139 (**magpi.cc/139**), for details on exactly where the various ports are and what they do.

Assembling the Raspberry Pi Case

If you're installing Raspberry Pi 5 in a case, this should be your first step. If you're using the Official Raspberry Pi Case, begin by splitting it into its three individual pieces: the red base, fan assembly and frame, and white lid.

Take the base and hold it so that the raised end is to your left and the lower end to your right.

Hold your Raspberry Pi 5, with no microSD card inserted, by its USB and Ethernet ports, at a slight angle. Gently lower the other side down into the base, so it looks like **Figure 5**. You should feel and hear a click as you seat it flat against the base.

Setting up the fan assembly

The fan should have arrived inserted into the fan assembly, and the fan assembly should already be inserted into its frame when you take it out of the box. If not, you can click it all together).

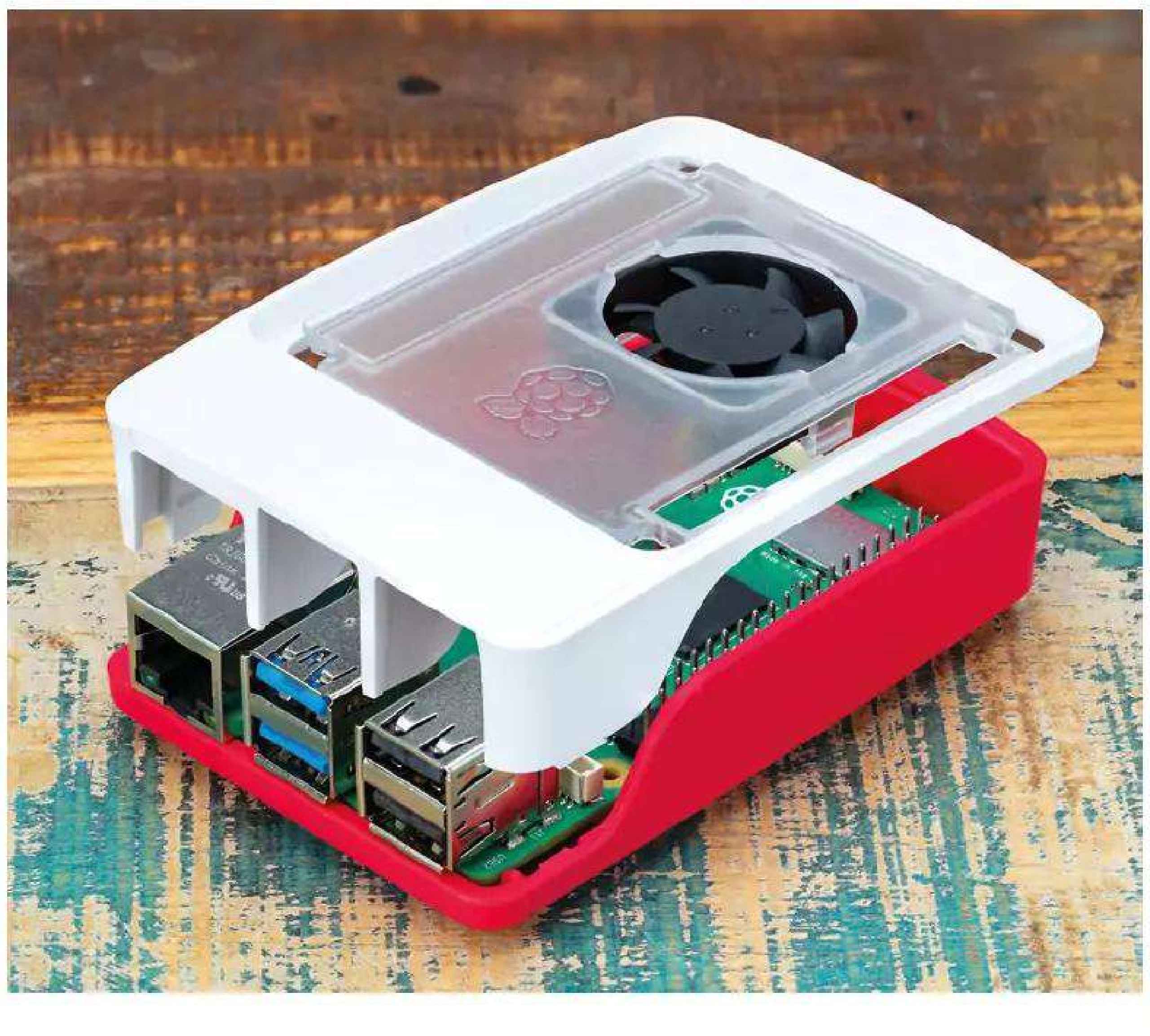


Figure 7: Attaching the fan assembly and frame

Next, plug the fan's white JST connector into the fan socket on the Raspberry Pi 5 as shown in **Figure 6**. It will only fit one way around, so you don't need to worry about connecting it backwards.

Click the fan assembly and frame into place as shown in **Figure 7**, and gently push down until you feel and hear a click.

If you want to cover everything in the case up, take the optional white lid and position it so that the Raspberry Pi logo is over the USB and Ethernet connectors on Raspberry Pi 5, as shown in **Figure 8**. To fasten it into place, gently push down on the middle of the lid until you hear a click.

Hats and lids

You can fit a HAT (Hardware Attached on Top) directly on top of Raspberry Pi 5 by removing the fan assembly, or you can stack it on top of the fan assembly and frame using 14mm-high standoffs and a 19mm GPIO extender. These will be available separately from authorised resellers.

Assembling the Raspberry Pi Zero Case

If you're installing Raspberry Pi Zero 2 W in a case, this should be your first step. If you're using the Official Raspberry Pi Zero Case, begin by unpacking it. You should have four pieces: a red base and three white lids.

If you're using Raspberry Pi Zero 2, you want to use the solid lid. If you're going to be using the GPIO header, pick the lid with the long rectangular

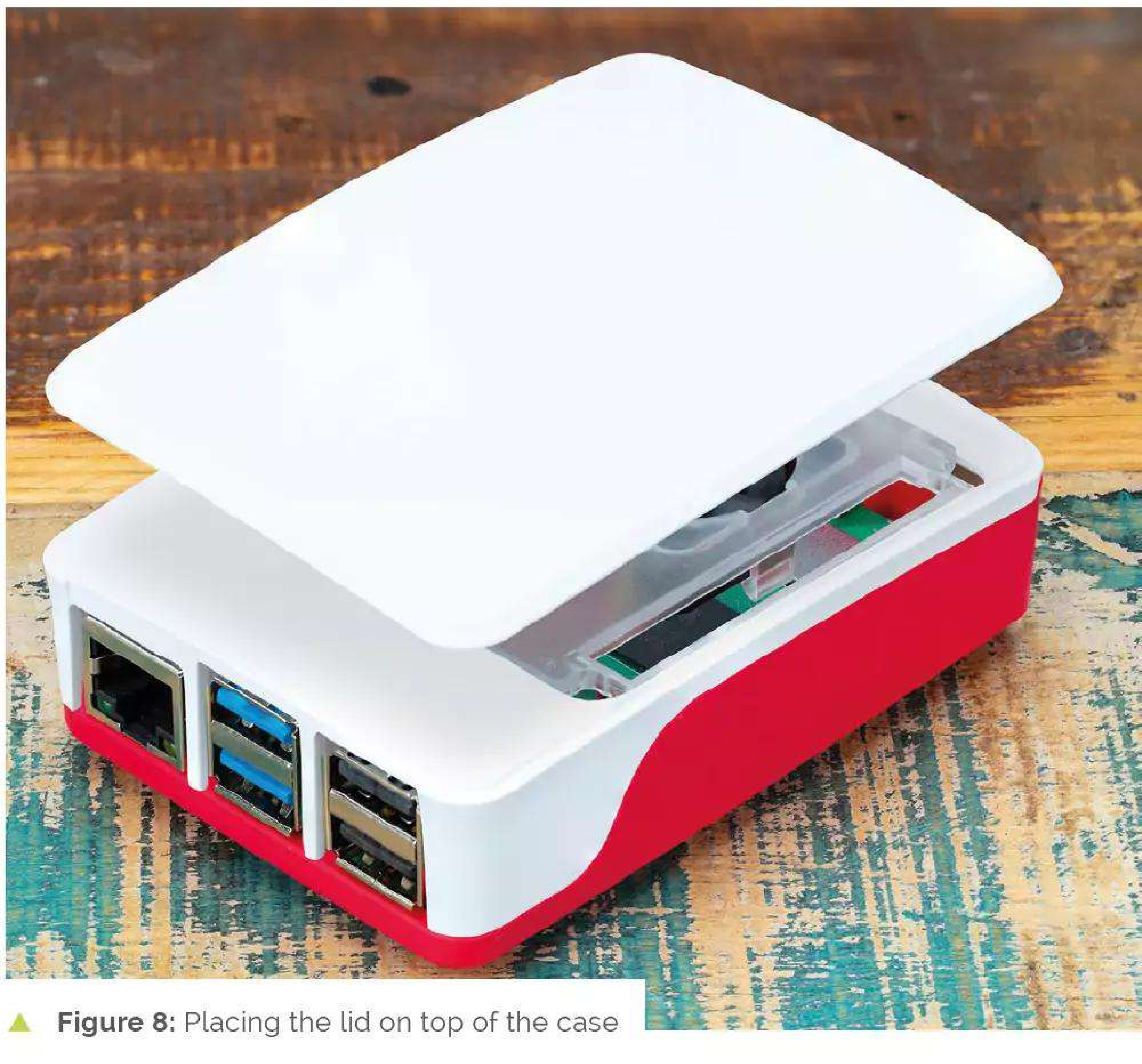
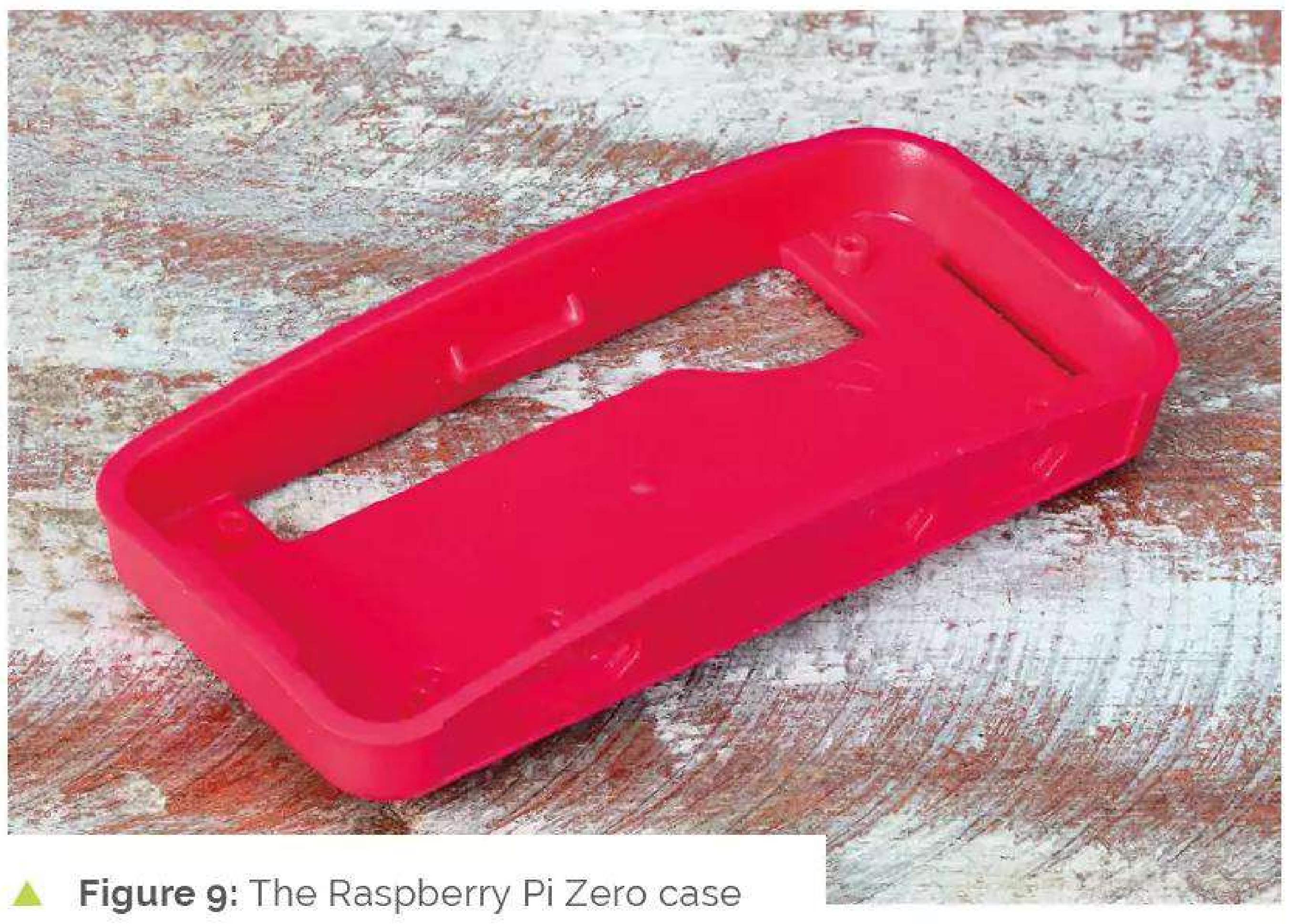
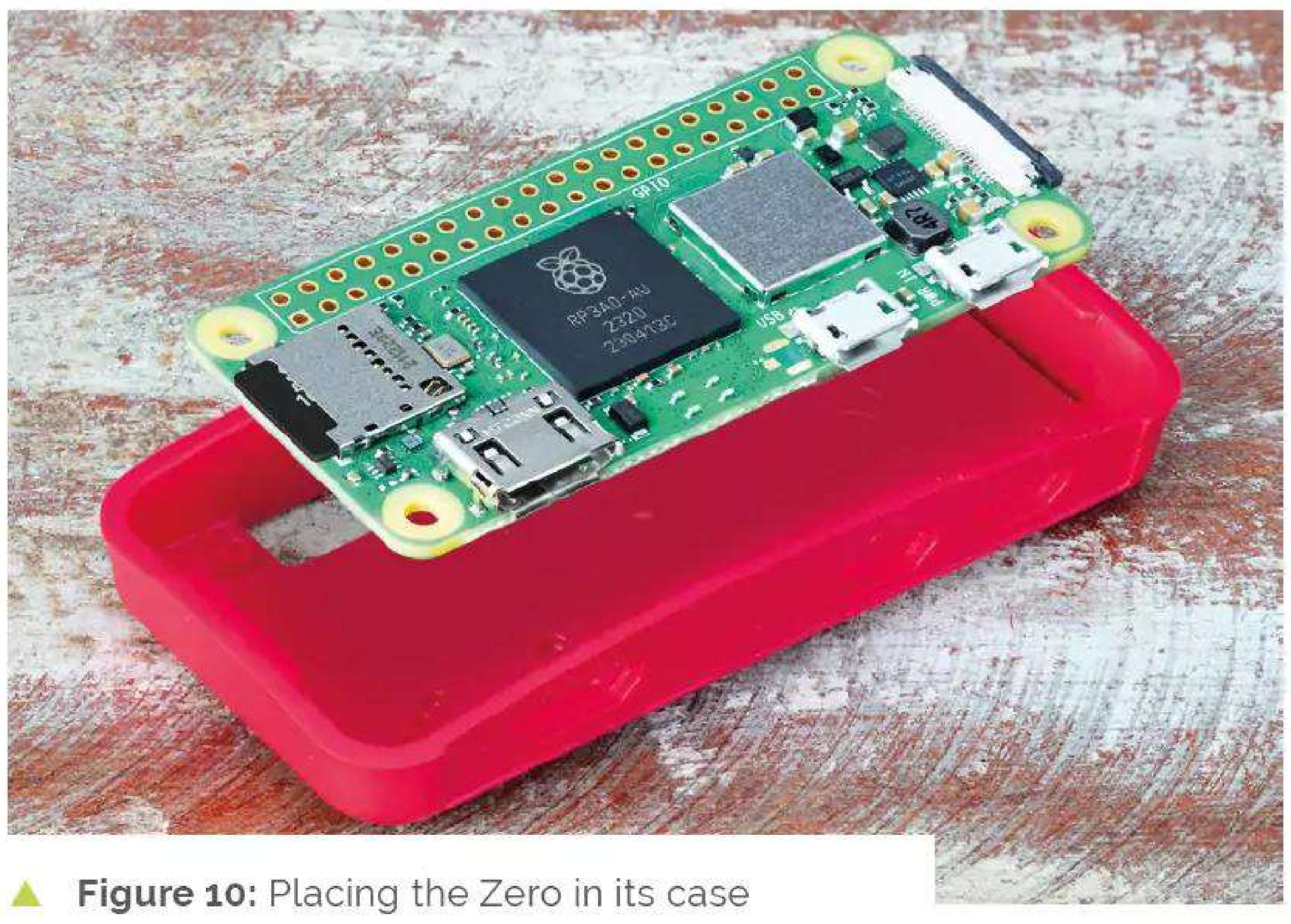


Figure 8: Placing the lid on top of the case



▲ **Figure 9:** The Raspberry Pi Zero case



▲ **Figure 10:** Placing the Zero in its case

> " The microSD card will slide in, then stop without a click "

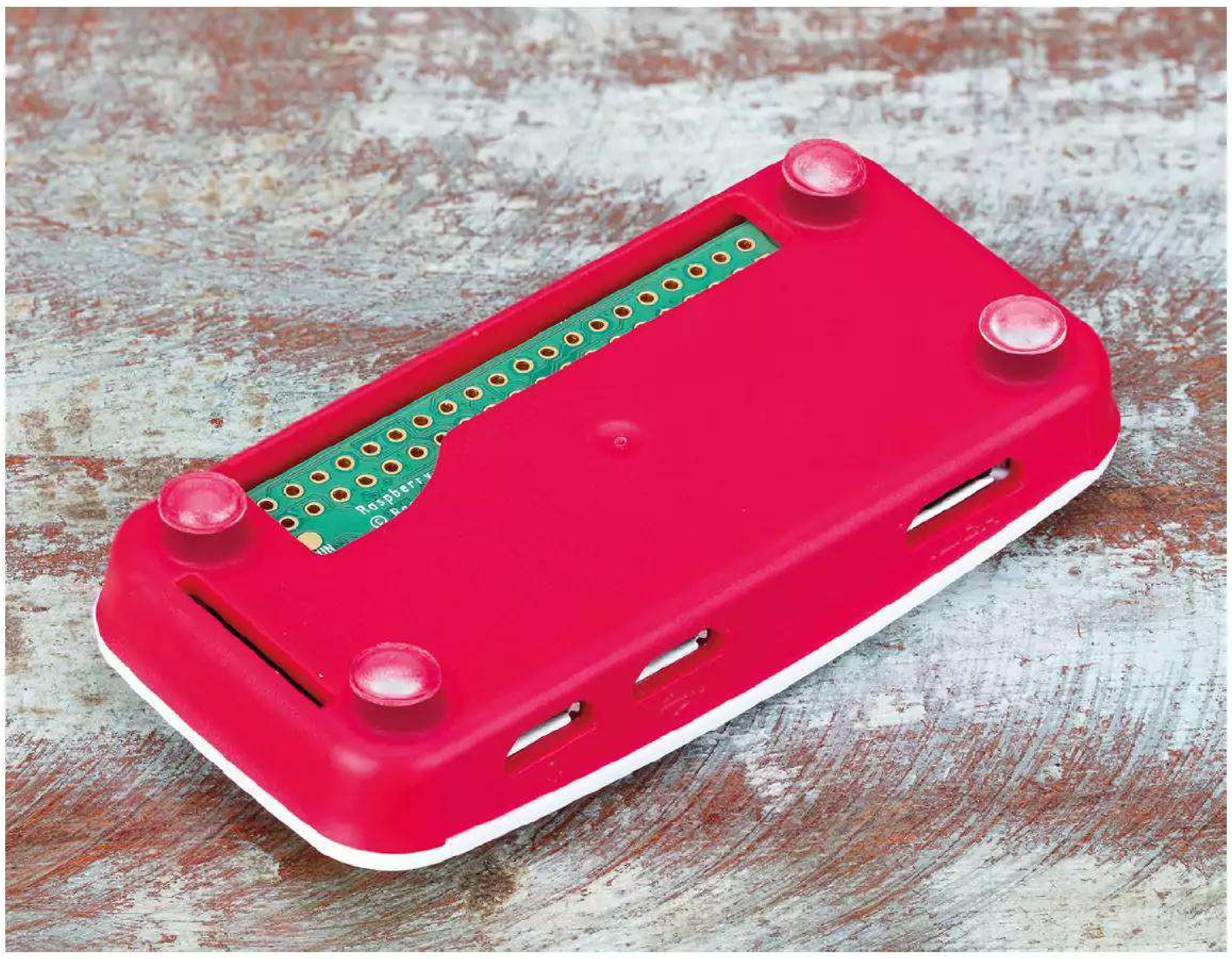
hole in it. If you have a Camera Module 1 or 2, pick the lid with the circular hole.

The Camera Module 3 and High Quality (HQ) Camera Module are not compatible with the Raspberry Pi Zero Case camera lid and must be used outside the case; there's a cut-out at the end of the Raspberry Pi Zero Case for the camera cable.

Take the base and place it flat on the table so that the cut-outs for the ports are facing towards you as shown in **Figure 9**.

Holding your Raspberry Pi Zero (with the microSD card inserted) by the edges of the board, line it up so the small circular posts in the corners of the base go into the mounting holes on the corners of Raspberry Pi Zero 2 W's circuit board. When they're lined up **Figure 10**), gently push Raspberry Pi Zero 2 W downwards until you hear a click and the ports are lined up with the cut-outs in the base.

Take your chosen white lid and place it on top of the Raspberry Pi Zero Case base, as shown in Chapter 11. If you're using the Camera Module lid, make sure that the cable isn't trapped. When the lid is in place, gently push it down until you hear a click.



▲ **Figure 12:** Attaching the feet

Camera Module and the Zero case

If you're using a Raspberry Pi Camera Module, use the lid with the circular hole. Line the Camera Module's mounting holes up with the cross-shaped posts in the lid, so that the camera's connector is facing towards the logo on the lid. Click it into place. Gently push the bar on the camera connector away from your Raspberry Pi, then push the narrower end of the included camera ribbon cable into the connector before pushing the bar back into place. Connect the wider end of the cable to the Camera Module in the same way.

At this point, you can also stick the included rubber feet to the bottom of the case (see **Figure 12**): flip it over, peel the feet off the backing sheet, and stick them in the circular indentations on the base to provide a better grip on your desk.

Connecting the microSD card

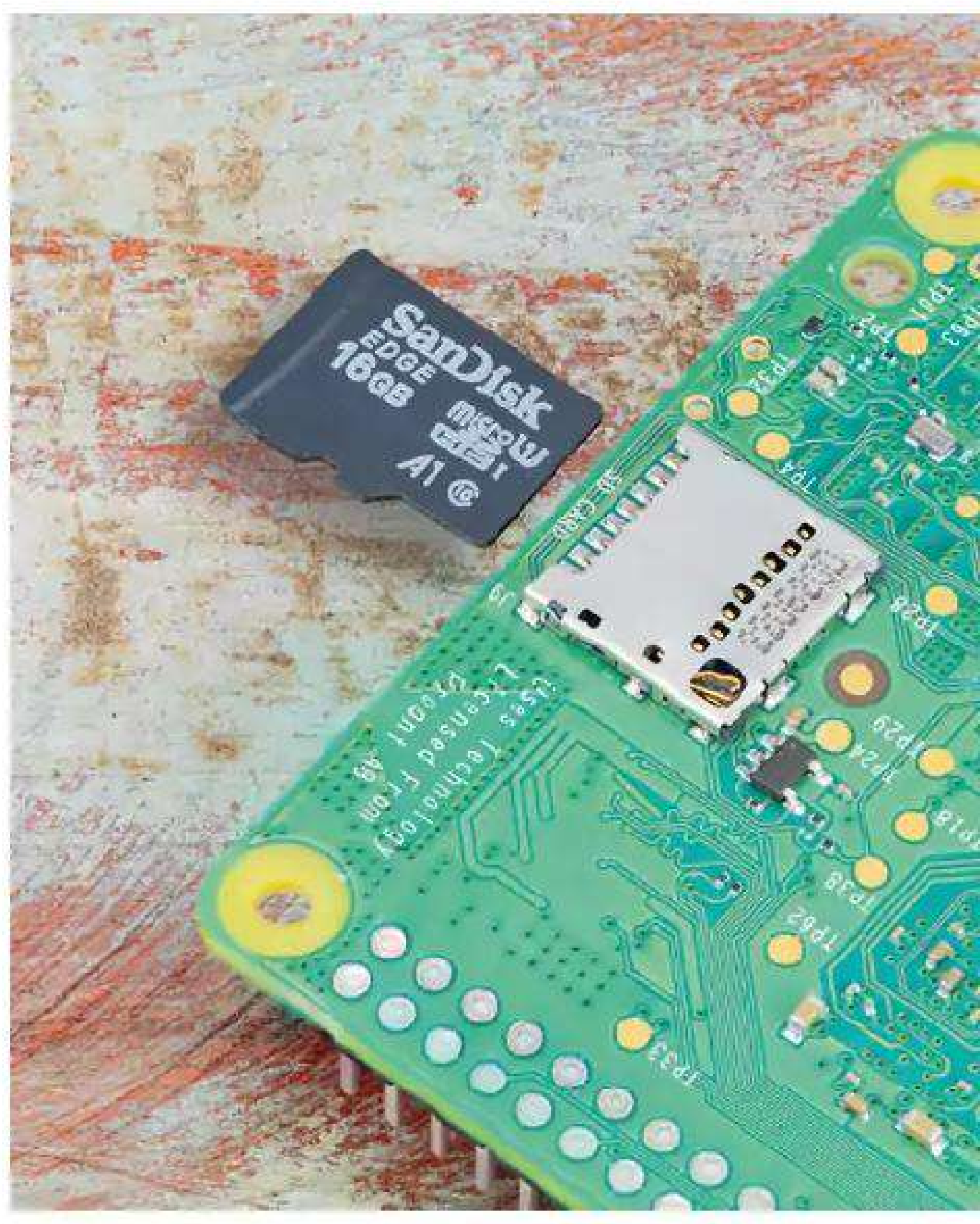
To install the microSD card, which is Raspberry Pi's storage, turn your Raspberry Pi (in its case if you're using one) upside-down and slide the card into the microSD slot with the card's label side facing away from Raspberry Pi. It will only fit in one way, and should slide home without too much pressure (see **Figure 13**).

The microSD card will slide into the socket connector, then stop without a click.

For Raspberry Pi Zero 2 W, the microSD slot is on the top at the left-hand side. Insert the card with the label facing away from your Raspberry Pi.

If you want to remove it again in the future, simply grip the end of the card and pull it gently out. If you're using an older model of Raspberry Pi, you'll need to give the card a gentle push first to unlock it; this isn't necessary with a Raspberry Pi 3, 4, 5, or any model of Raspberry Pi Zero.

▼ **Figure 13:** Inserting the microSD card



Connecting a keyboard and mouse

Connect the keyboard's USB cable to any of the four USB ports (black USB 2.0 or blue USB 3.0) on Raspberry Pi as shown in **Figure 14** If you're using the Official Raspberry Pi Keyboard, there's a USB port on its back for the mouse. Otherwise, just connect the USB cable from your mouse to another USB port on Raspberry Pi.

For Raspberry Pi Zero 2 W, you'll need to use a micro-USB OTG adapter cable. Insert this into the left-hand micro-USB port, then connect the USB cable from your keyboard to the USB OTG adapter.

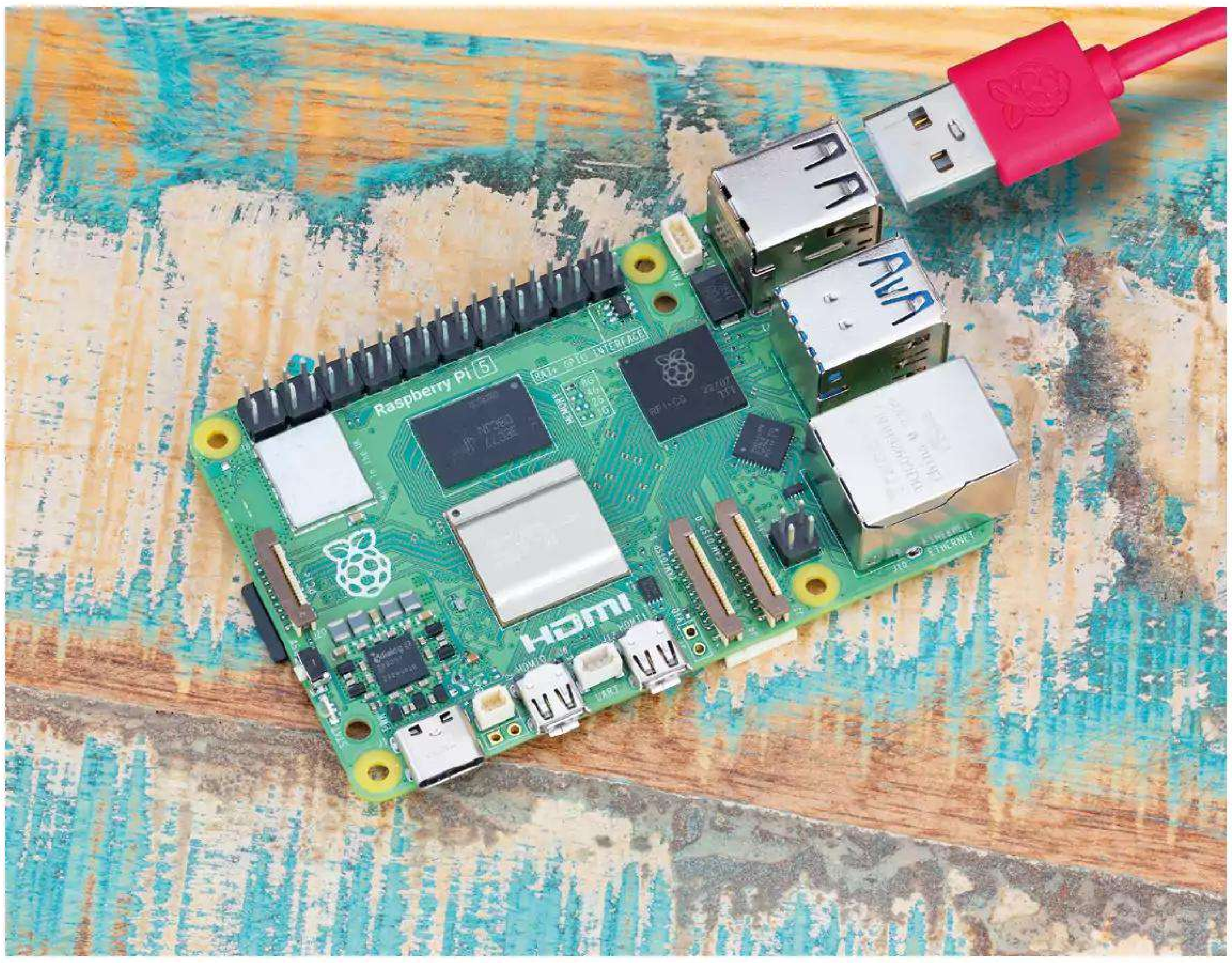
If you are using a keyboard with a separate mouse, rather than one with a built-in touchpad, you'll also need to use a powered USB hub. Connect the micro-USB OTG adapter cable as above, then connect the hub's USB cable to the USB OTG adapter before connecting your keyboard and mouse to the USB hub. Finally, connect the hub's power adapter and switch it on.

The USB connectors for the keyboard and mouse should slide home without too much pressure; if you're having to force the connector in, there's something wrong. Check that the USB connector is the right way up!

Keyboard and mouse

The keyboard and mouse act as your main means of telling Raspberry Pi what to do; in computing, these are known as input devices, in contrast with the display, which is an output device.

▼ **Figure 14:** Plugging a USB cable into a Raspberry Pi 5

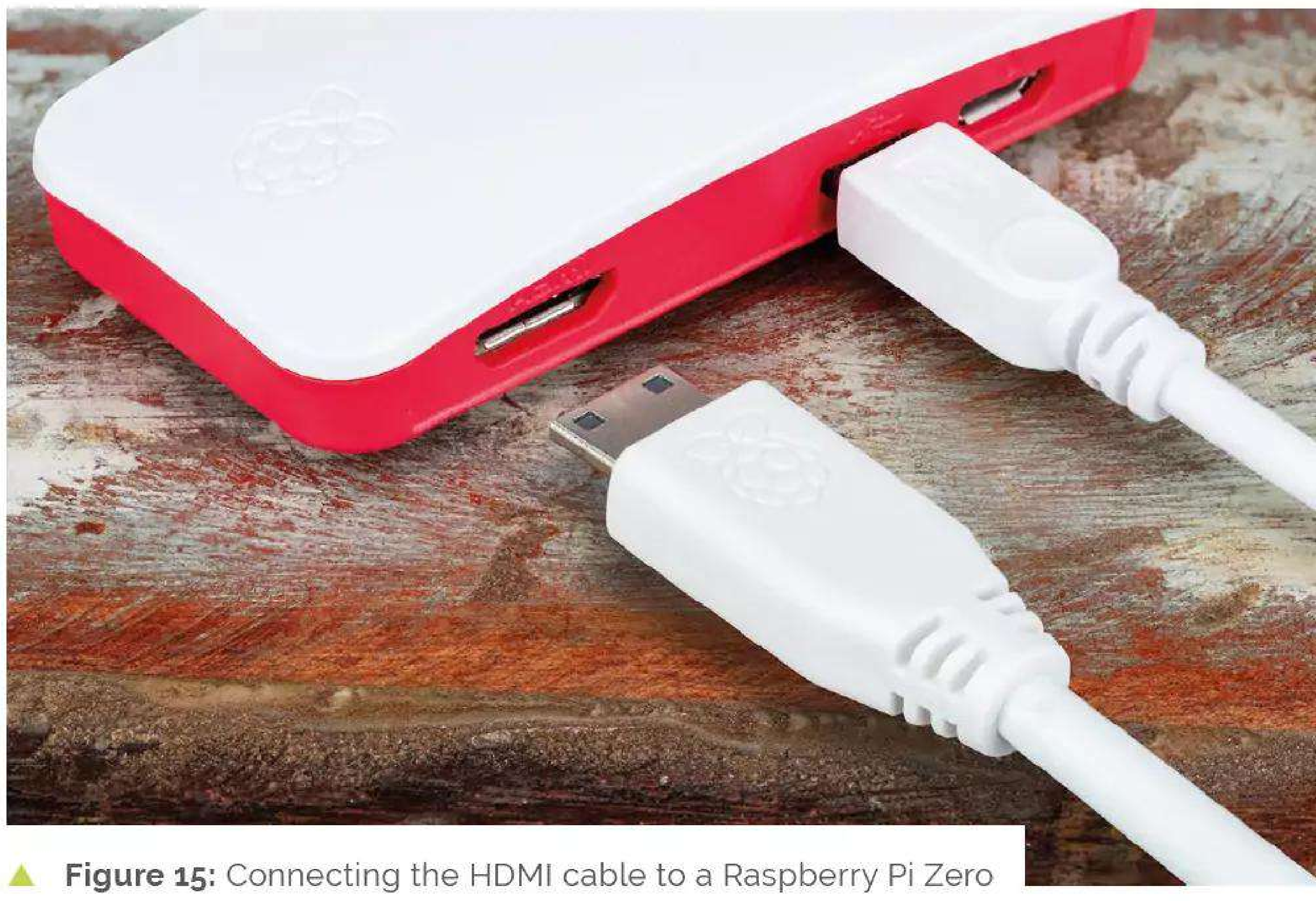


Connecting a display

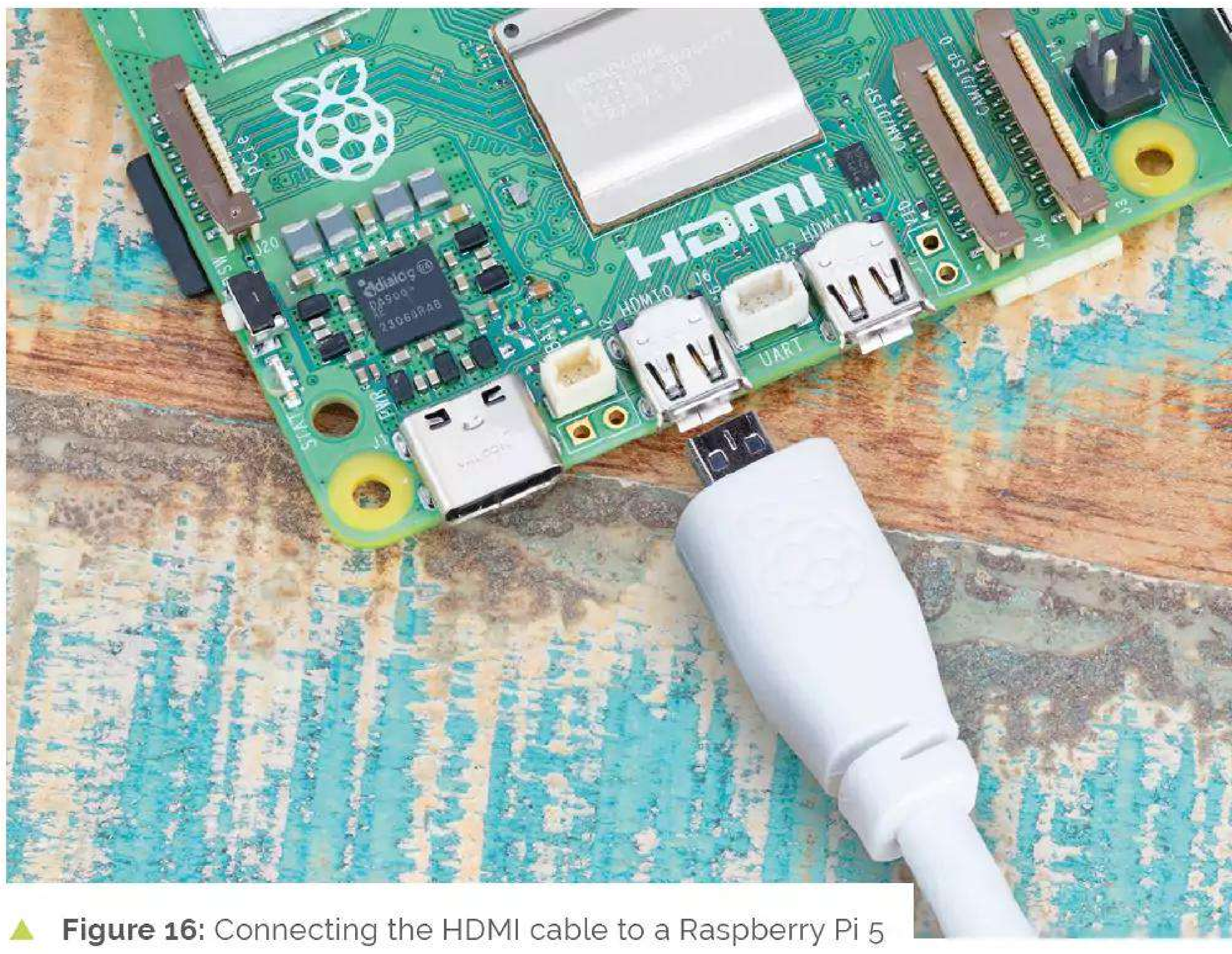
For Raspberry Pi 4 and Raspberry Pi 5, take the micro-HDMI cable and connect the smaller end to the micro-HDMI port closest to the USB Type-C port on your Raspberry Pi. Connect the other end to your display as shown in **Figure 15**.

For Raspberry Pi Zero 2 W (**Figure 16**), take the mini-HDMI cable and connect the smaller end to the mini-HDMI port at the left-hand side of your Raspberry Pi, under the microSD slot. The other end connects to your display.

If your display has more than one HDMI port, look for a port number next to the connector itself; you'll need to switch the screen to this input to see your Raspberry Pi's display. If you can't see a port number, don't worry, you can simply switch through each input in turn until you find Raspberry Pi.



▲ **Figure 15:** Connecting the HDMI cable to a Raspberry Pi Zero



▲ **Figure 16:** Connecting the HDMI cable to a Raspberry Pi 5

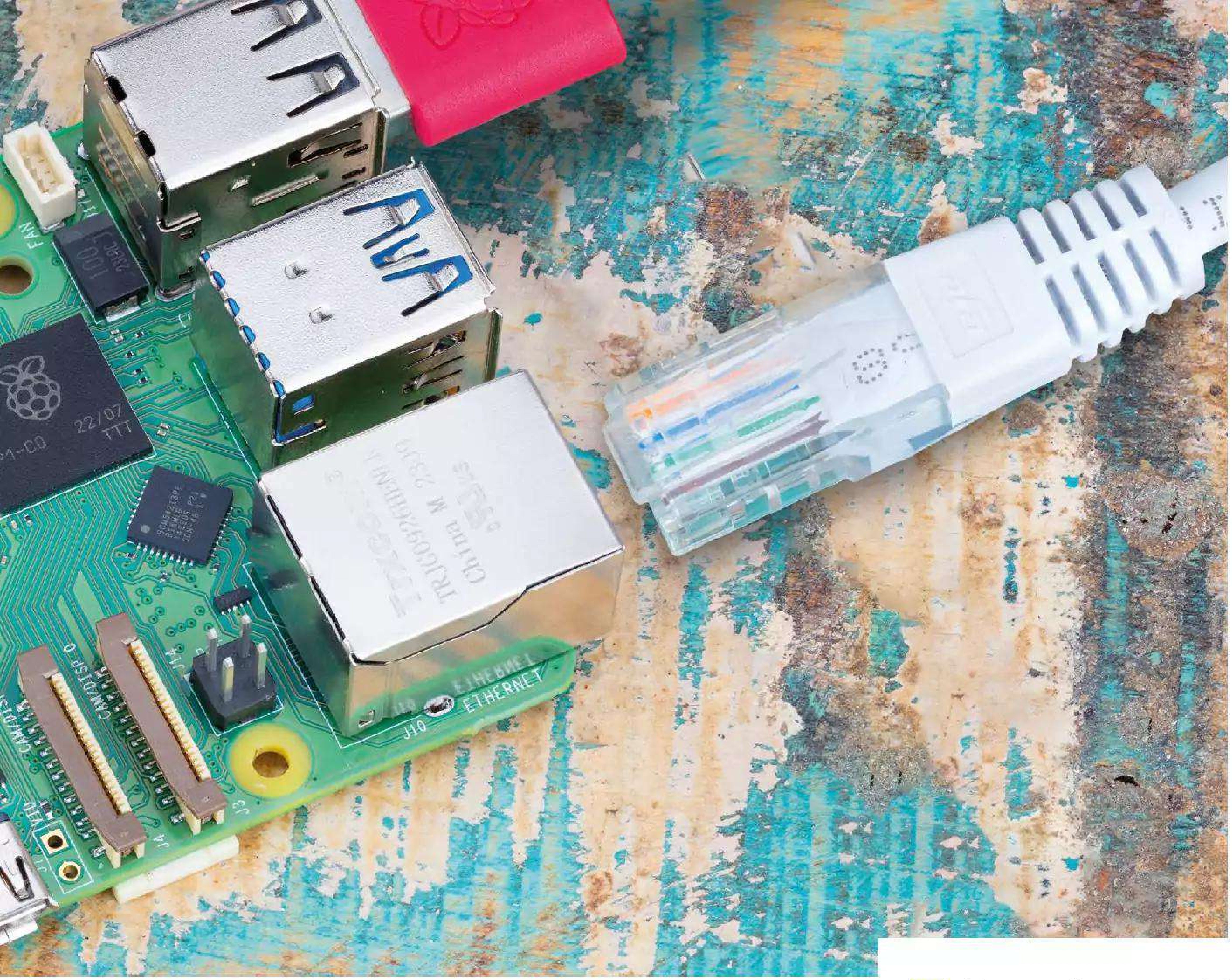


Figure 17: Connecting Raspberry Pi 5 to Ethernet

> " Raspberry Pi will turn on as soon as it's connected to a live power supply "

TV connection

If your TV or monitor doesn't have an HDMI connector, that doesn't mean you can't use a Raspberry Pi. Adapter cables, available from any electronics stockist, will allow you to convert the micro or mini HDMI port on your Raspberry Pi to DVI-D, DisplayPort, or VGA for use with other computer monitors.

Connecting a network cable (optional)

To connect your Raspberry Pi to a wired network, take a network cable – known as an Ethernet cable – and push it into Raspberry Pi's Ethernet port, with the plastic clip facing downwards, until you hear a click (see **Figure 17**). If you need to remove the cable, squeeze the plastic clip inwards towards the plug and gently slide the cable free again.

The other end of your network cable should be connected to any free port on your network hub, switch, or router in the same way.

Connecting a power supply

Connecting your Raspberry Pi to a power supply is the final step in the hardware setup process. It's the last thing you'll do before you start setting up its software. Your Raspberry Pi will turn on as soon as it's connected to a live power supply.

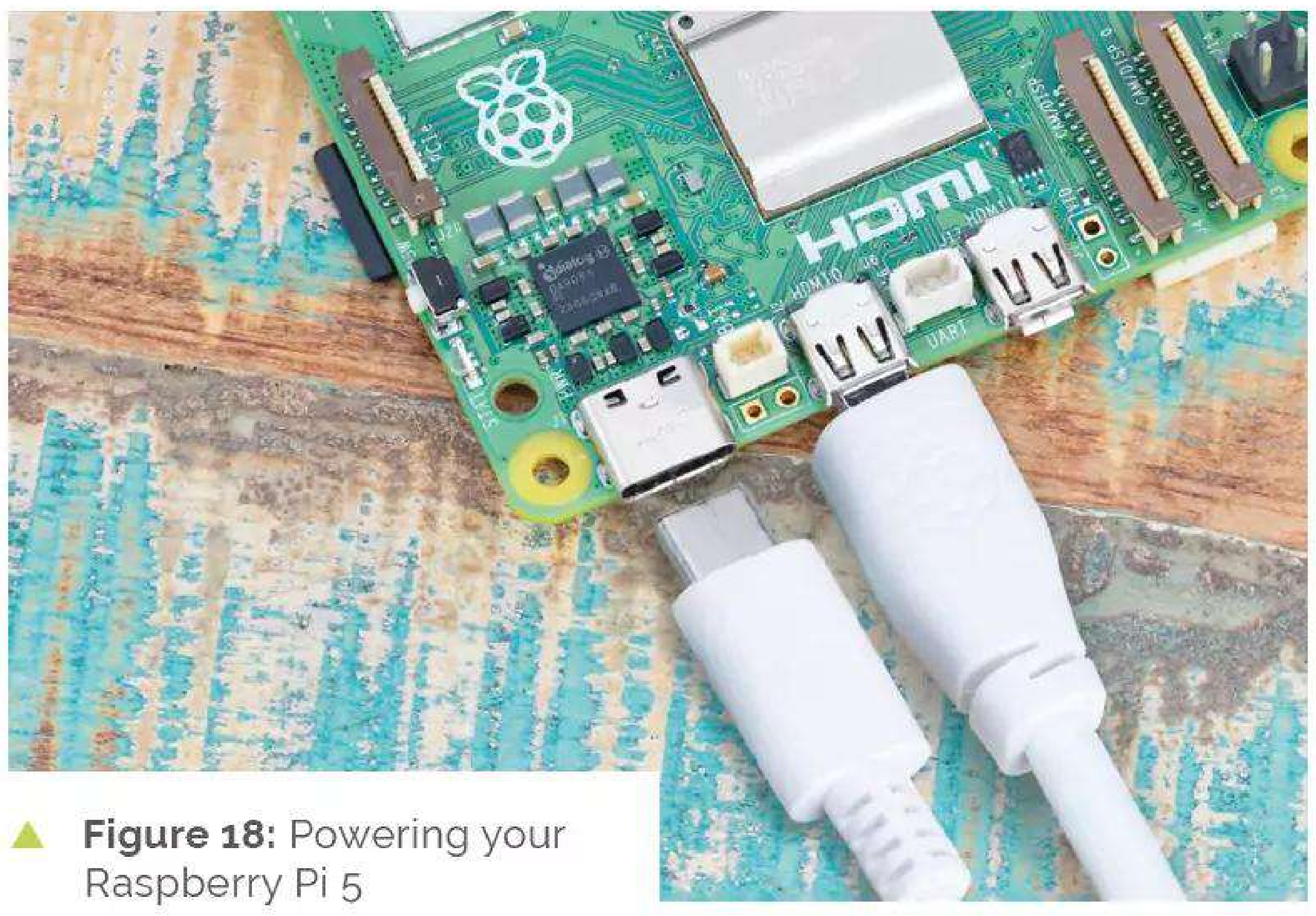


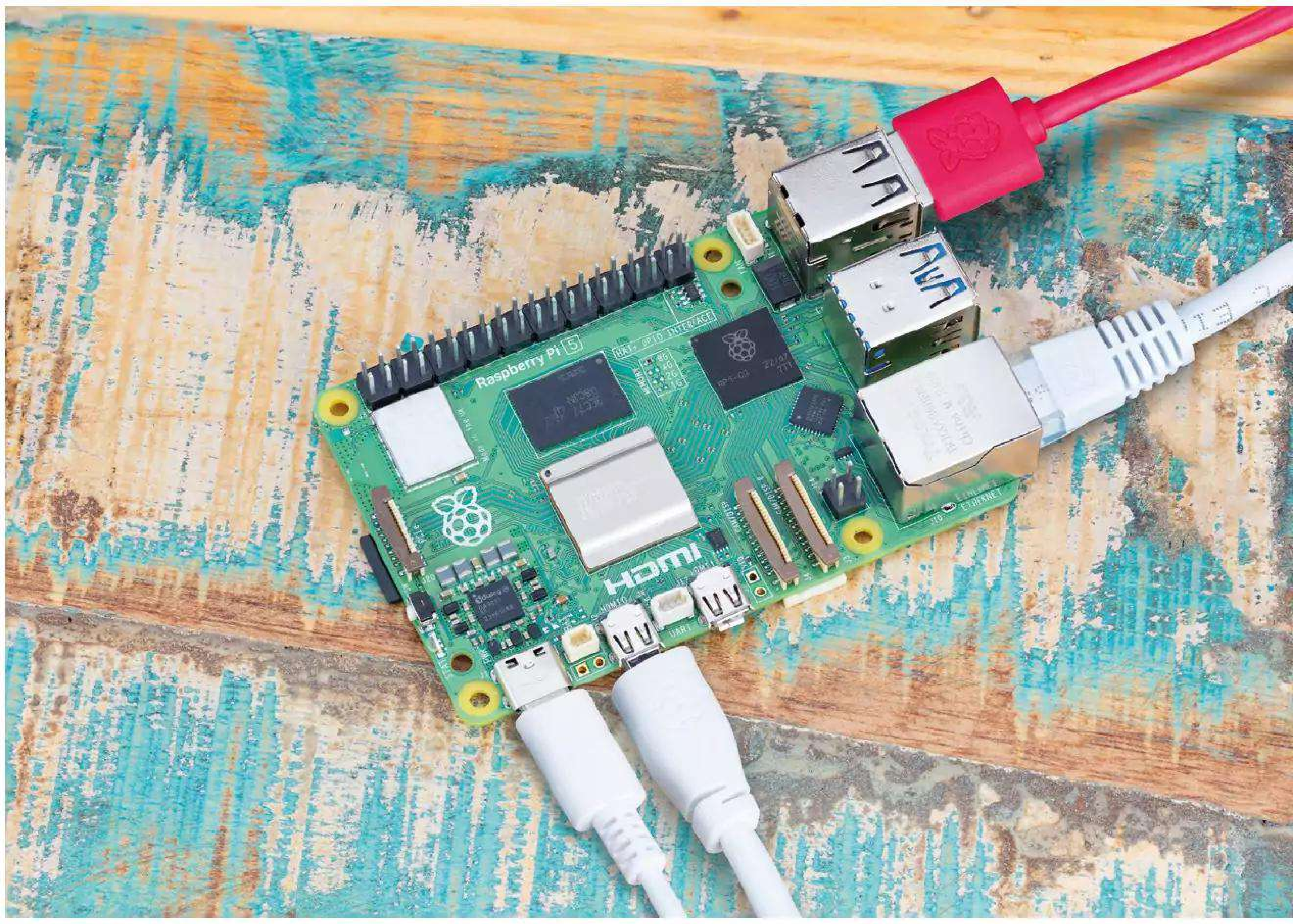
Figure 18: Powering your Raspberry Pi 5

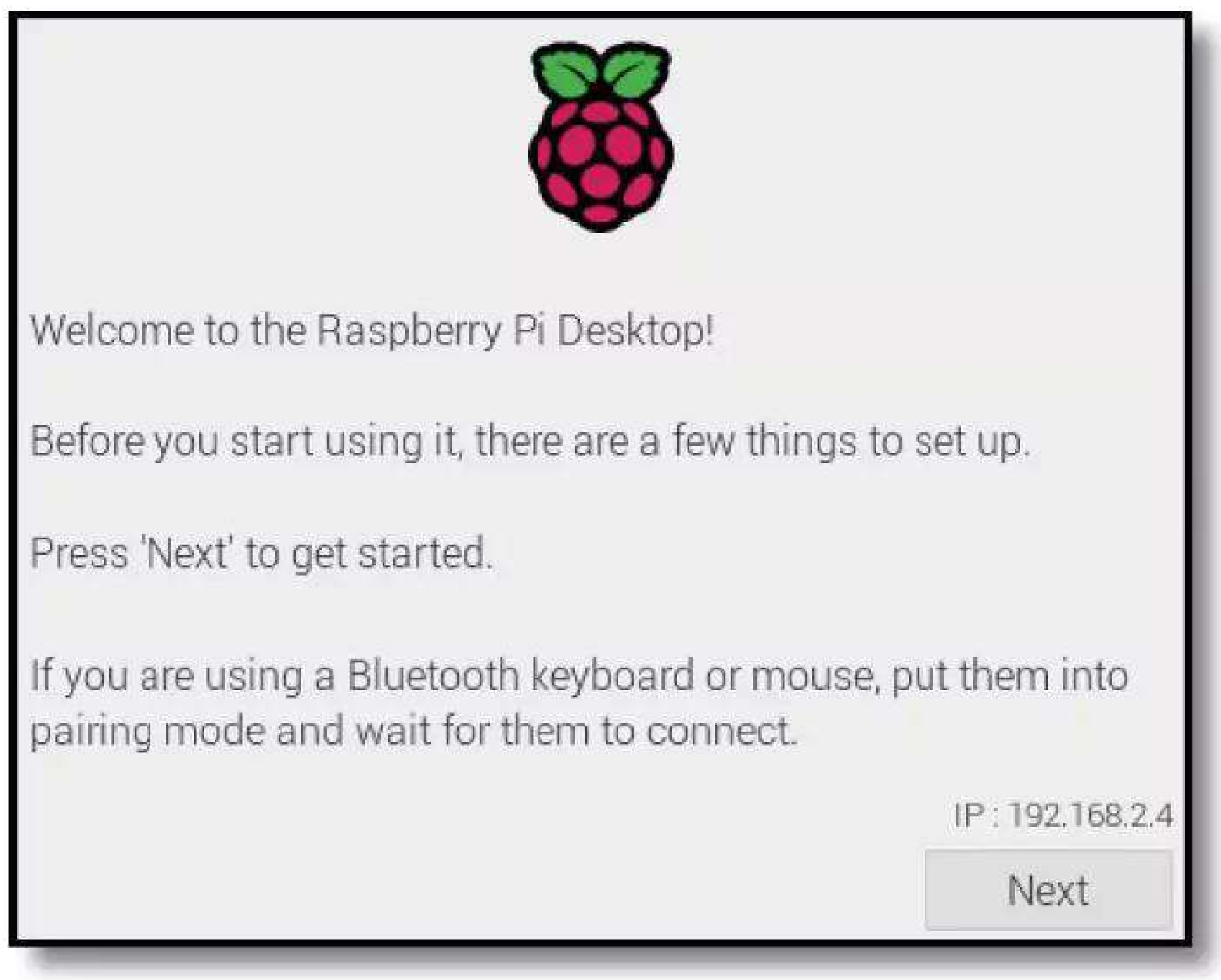
For Raspberry Pi 4 and Raspberry Pi 5, connect the USB-C end of the power supply cable to the USB-C power connector on your Raspberry Pi as in **Figure 18**. It can go in either way around, and should slide home gently. If your power supply has a detachable cable, make sure the other end is plugged into the body of the power supply.

For Raspberry Pi Zero 2 W, connect the micro-USB end of the power supply cable to the right-hand micro-USB port on your Raspberry Pi. It can only go in one way up, so make sure to check its orientation before pushing it gently in.

Congratulations: you have put your Raspberry Pi together (**Figure 19**)!

Figure 19: Your Raspberry Pi is ready to go





Welcome to the Raspberry Pi Desktop!

Before you start using it, there are a few things to set up.

Press 'Next' to get started.

If you are using a Bluetooth keyboard or mouse, put them into pairing mode and wait for them to connect.

IP : 192.168.2.4

▶ **Figure 20:** The Raspberry Pi OS Welcome Wizard

Finally, connect the power supply to a mains socket, switch the socket on, and your Raspberry Pi will immediately start running.

You'll briefly see a rainbow-coloured cube followed by an informational screen with a Raspberry Pi logo on it. You may also see a blue screen appear as the operating system resizes itself to make full use of your microSD card. If you see a black screen, wait a few minutes; the first time Raspberry Pi boots it will perform some housekeeping in the background, which can take a little time.

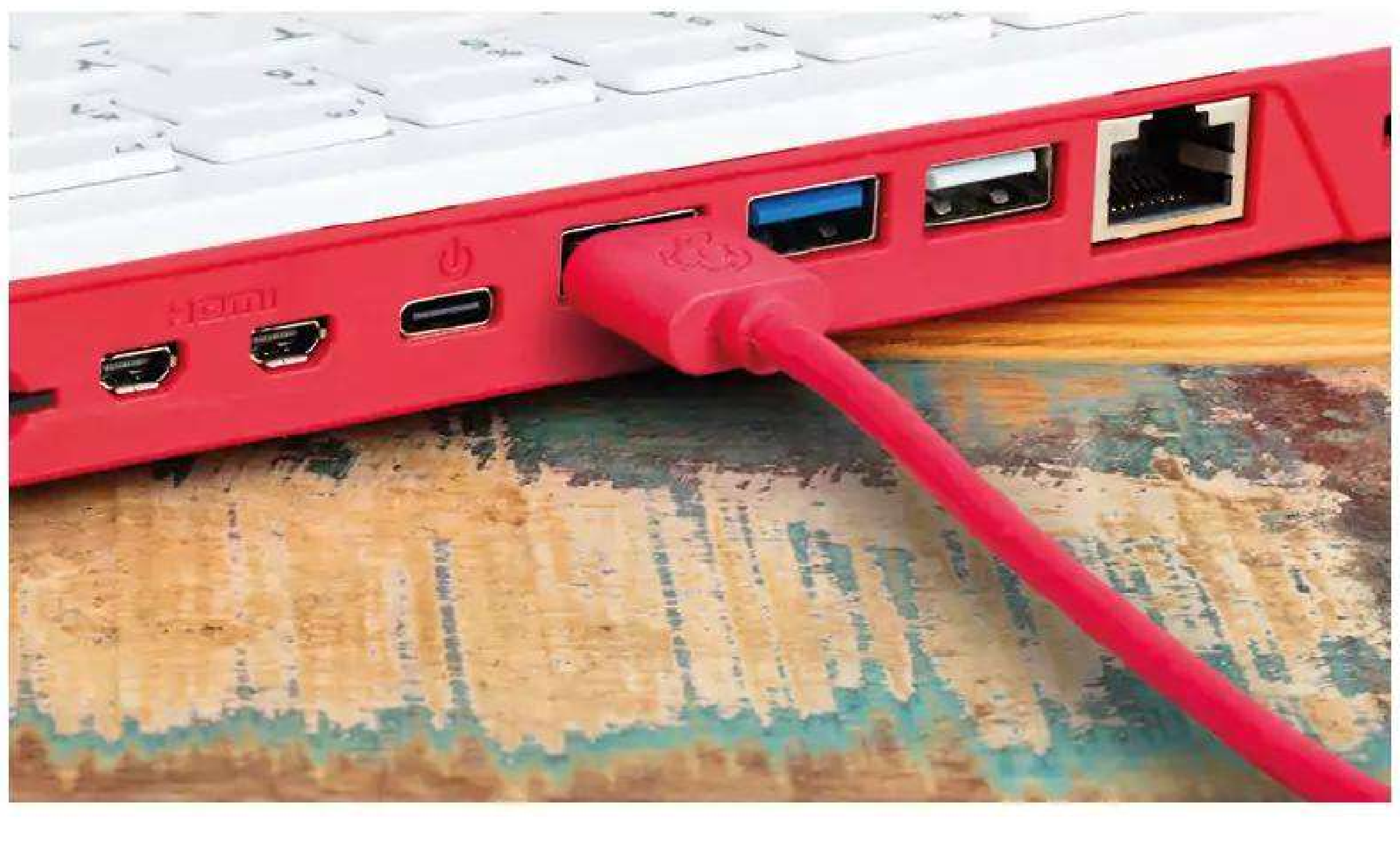
After a while you'll see the Raspberry Pi OS Welcome Wizard (**Figure 20**). Your operating system is now ready to be configured.

Setting up Raspberry Pi 400

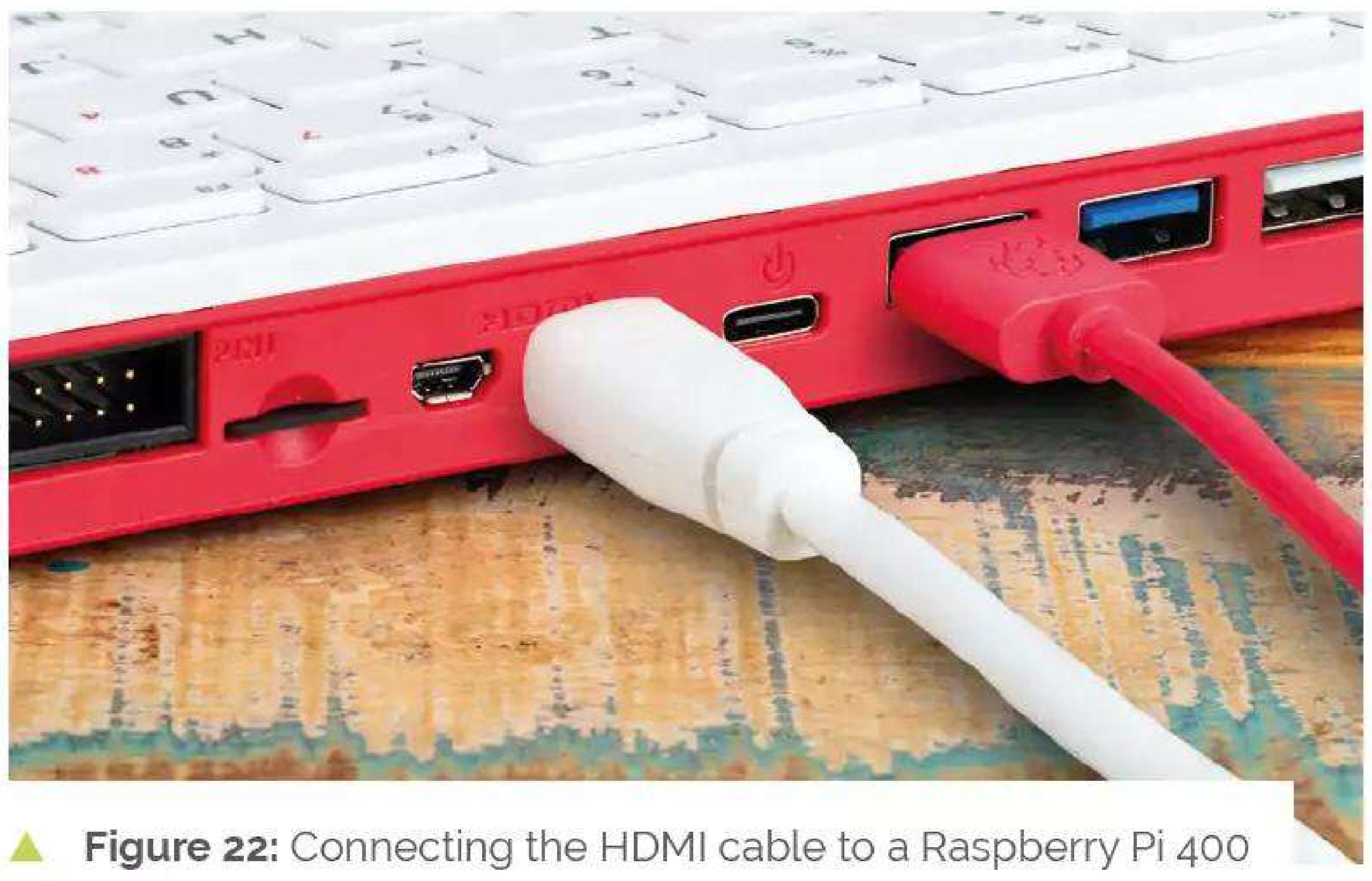
Unlike Raspberry Pi 4, Raspberry Pi 400 comes with a built-in keyboard and the microSD card already installed. You'll still need to connect a few cables to get started, but it should only take you a few minutes.

Connecting a mouse

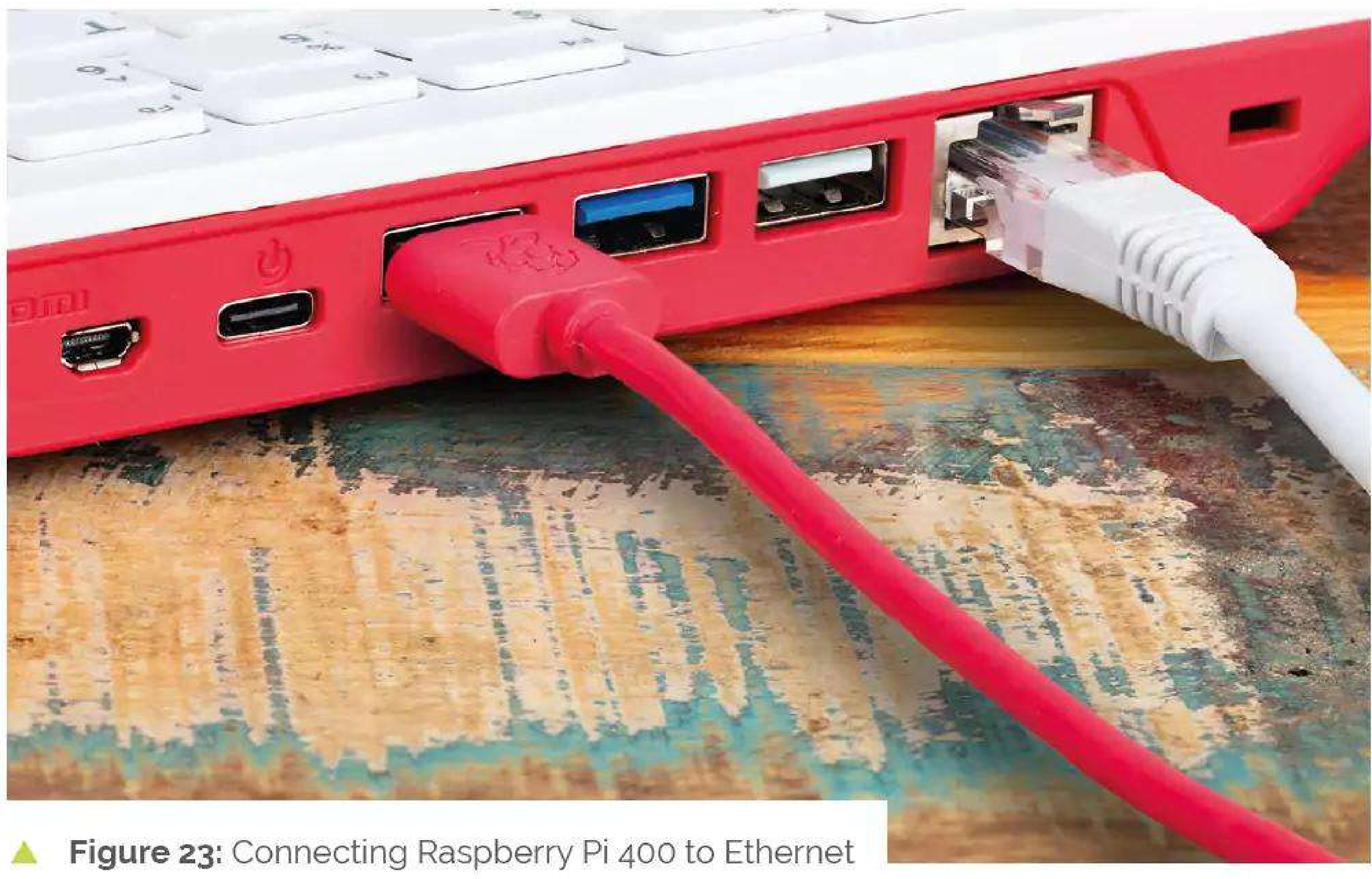
Raspberry Pi 400's keyboard is already connected, leaving you with just the mouse to add. Take the USB cable at the end of the mouse and insert it into any of the three USB ports (2.0 or 3.0) on the rear panel of your Raspberry Pi 400. If you want to save the two blue, high-speed USB 3.0 ports for other accessories, use the white USB 2.0 port.



▶ **Figure 21:** Plugging a USB cable into a Raspberry Pi 400



▲ **Figure 22:** Connecting the HDMI cable to a Raspberry Pi 400



▲ **Figure 23:** Connecting Raspberry Pi 400 to Ethernet

The USB connector should slide home without too much pressure (see **Figure 21**). If you're having to force the connector in, there's something wrong. Check that the USB connector is the right way up!

Connecting a display

Take the micro-HDMI cable and connect the smaller end to the micro -HDMI port closest to the microSD slot on your Raspberry Pi 400, and the other end to your display, as shown in **Figure 22**. If your display has more than one HDMI port, look for a port number next to the connector itself; you'll need to switch the TV or monitor to this input to see Raspberry Pi's display. If you can't see a port number, don't worry: you can simply switch through each input in turn until you find Raspberry Pi.

Connecting a network cable (optional)

To connect your Raspberry Pi 400 to a wired network, take a network cable – known as an Ethernet cable – and push it into Raspberry Pi 400's Ethernet port, with the plastic clip facing upwards, until you hear a click (**Figure 23**). If you

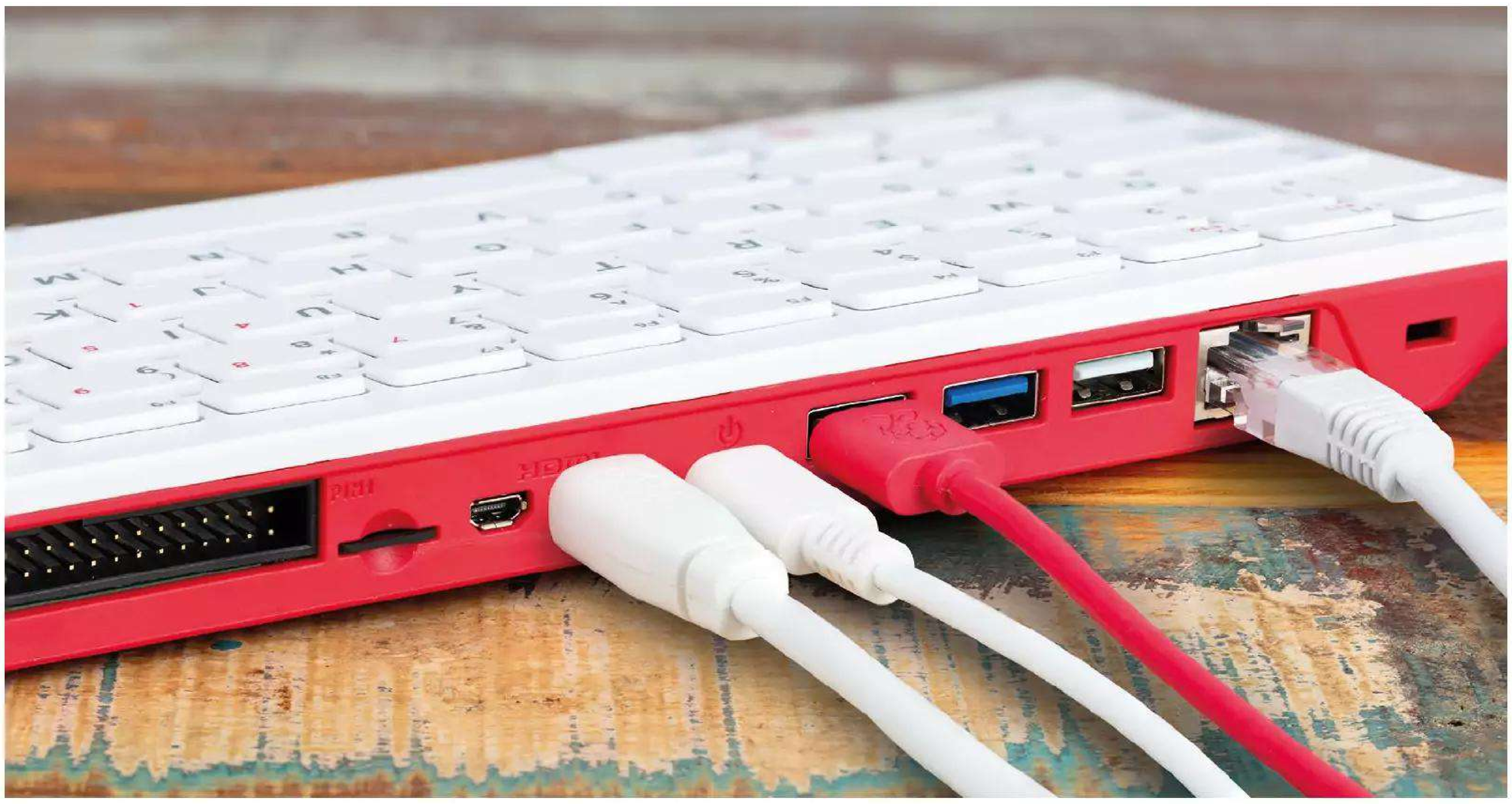


Figure 24: Your Raspberry Pi 400 is all wired up!

need to remove the cable, just squeeze the plastic clip inwards toward the plug and gently slide the cable free again. The other end of your network cable should be connected to any free port on your network hub, switch, or router in the same way.

Connecting a power supply

Connecting Raspberry Pi 400 to a power supply is the very last step in the hardware setup process, and it's one you should do only when you're ready to set up its software. Raspberry Pi 400 does not have a power switch and will turn on as soon as it is connected to a live power supply.

First, connect the USB Type-C end of the power supply cable to the USB Type-C power connector on Raspberry Pi. It can go in either way around and should slide home gently. If your power supply has a detachable cable, make sure the other end is plugged into the body of the power supply.

Finally, connect the power supply to a mains socket and switch the socket on: your Raspberry Pi 400 will immediately start running. Congratulations, you have put your Raspberry Pi 400 together (**Figure 24**)!

You'll briefly see a rainbow-coloured cube followed by an informational screen with a Raspberry Pi logo on it. You may also see a blue screen appear as the operating system resizes itself to make full use of your microSD card. If you see a black screen, wait a few minutes: the first time Raspberry Pi boots it will perform some housekeeping in the background, which can take a little time.

After a while you'll see the Raspberry Pi OS Welcome Wizard. Your operating system is now ready to be configured.

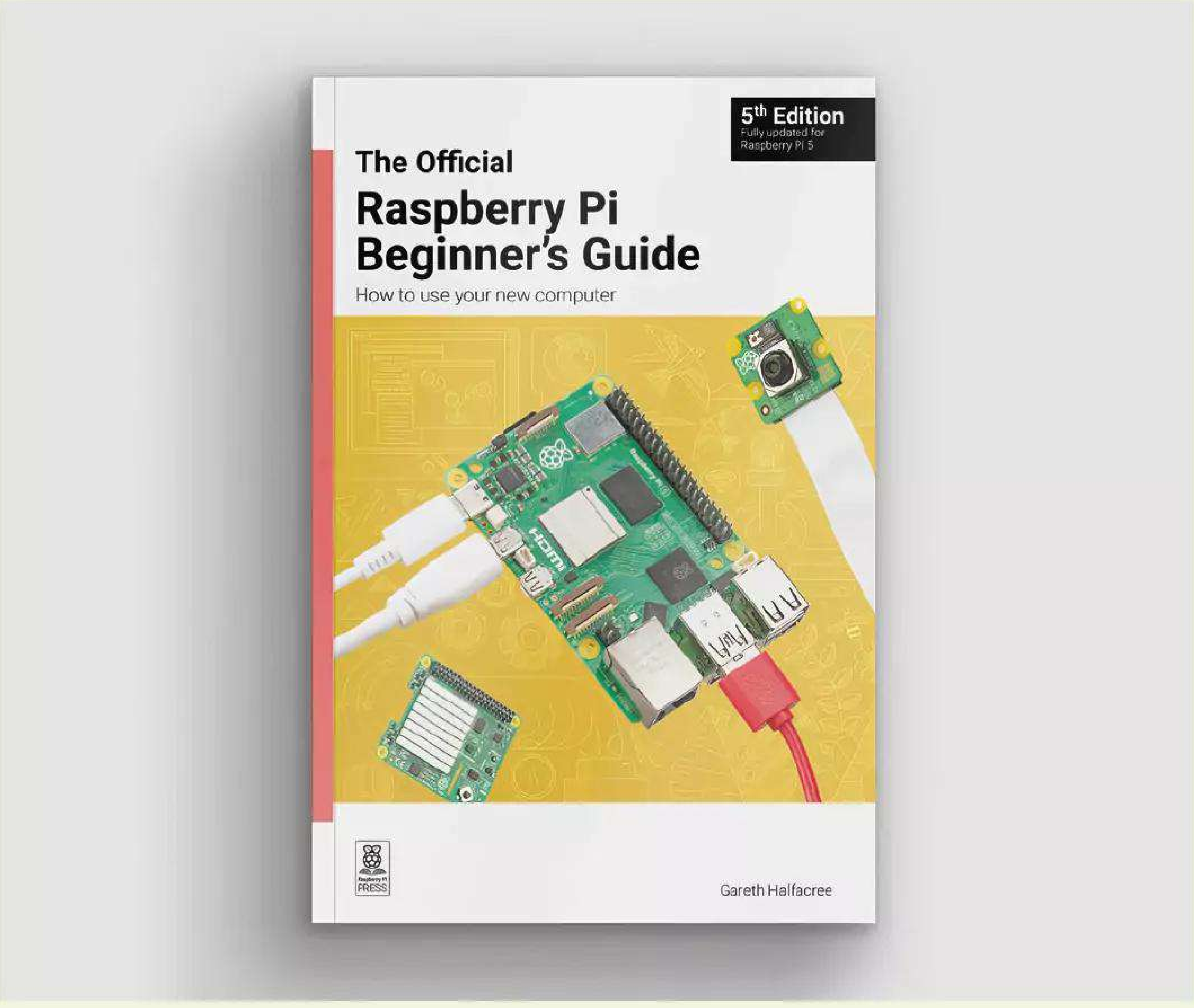
The Official Raspberry Pi Beginner's Guide

This tutorial is taken from *The Official Raspberry Pi Beginner's Guide 5th Edition*. This 290-page book is packed with guidance on how to set up your Raspberry Pi, install its operating system, and start using this fully functional computer.

With this book, you can start coding projects, with step-by-step guides using the Scratch 3, Python, and MicroPython programming languages.

Plus you'll experiment with connecting electronic components, and have fun creating amazing projects. You can buy a copy from our online store, or read it in the Bookshelf app in Raspberry Pi OS.

magpi.cc/beginnersguide



Part 01

Read floppy disks and CD-ROMs with Raspberry Pi 5

Play old games from original media, back up your floppies before they die, and image replacement disks

MAKER

K.G. Orphanides

While researching this feature, K.G. ddrescued their first published essay, and still hasn't stopped cringing about it. Beware that which you might raise up.

twoot.space/@owlbear

You'll Need

- Raspberry Pi 5
- USB disc drive (e.g. Asus SBC-06D2X-U) **magpi.cc/asusbluray**
- USB floppy drive (e.g. DELL MPF82E) **magpi.cc/dellfloppy**

Raspberry Pi 5 distributes enough power via its Universal Serial Bus to run most modern USB disc drives correctly. This is in contrast to our last attempt, in 2020 with Raspberry Pi 4, when we couldn't get modern dual-USB DVD and Blu-ray drives to power up correctly. Back then, in our CD-ROM console tutorial (**magpi.cc/cdromconsole**), we used a SATA DVD drive and USB adaptor kit. That approach still has its uses, but for most purposes, a modern external disc drive is now more convenient. We'll also look at the current state of USB floppy drive support and show you how to create a single mount point for disk swapping.

01 Connect your drive

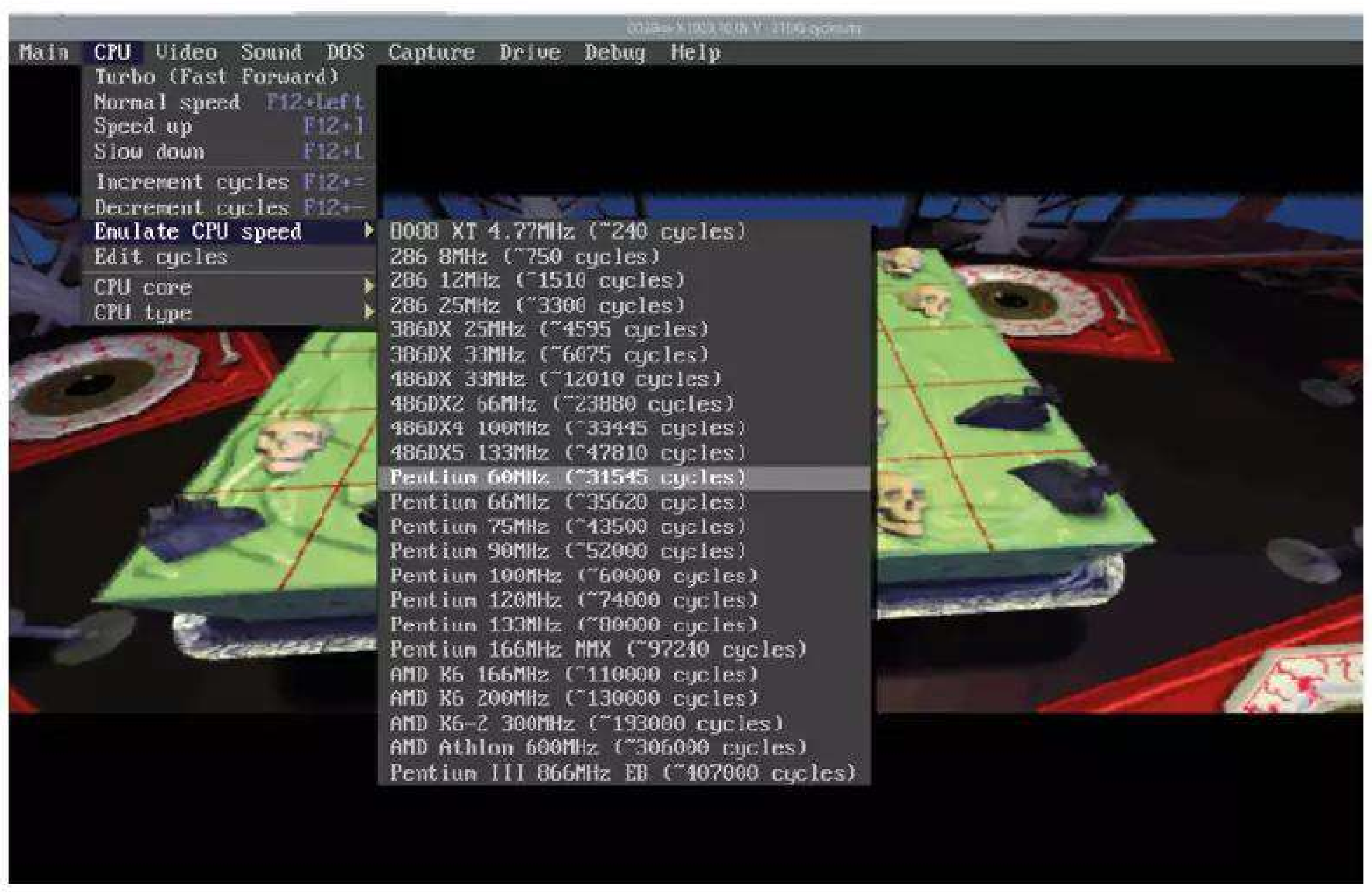
We used an Asus SBC-06D2X-U CD/DVD/Blu-ray writer drive throughout this tutorial, with both of its USB-A connectors hooked up to Raspberry Pi's USB3 ports. You can find them for around £100, but any similar drive should work. Using a powered USB hub can produce unexpected issues with read speeds, so it's best to use Raspberry Pi 5's ports.

02 Standard use cases

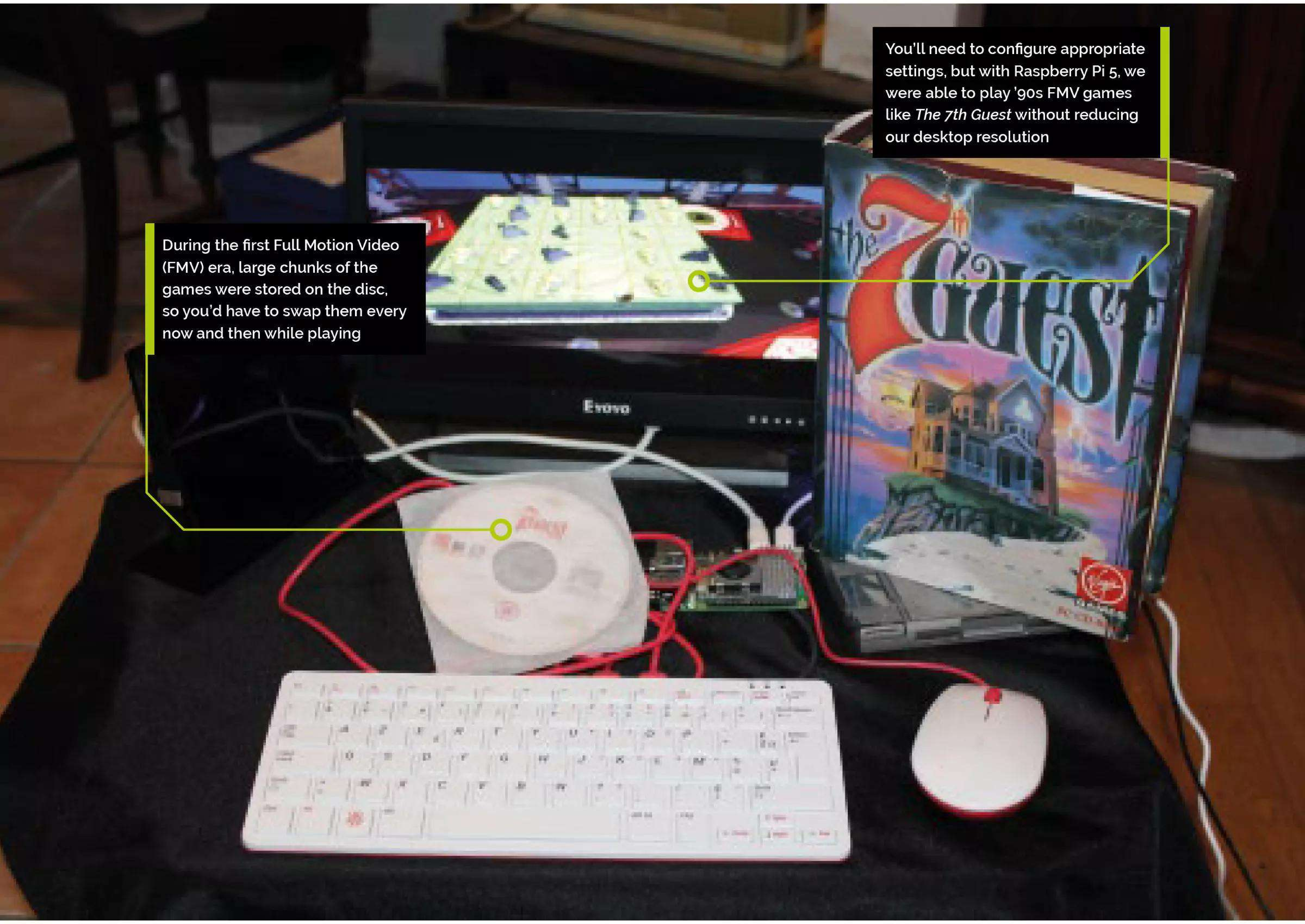
We're shortly going to look at emulation-friendly approaches to disc mounting, but if you're just using discs to read data, burn a CD, play movies, or install modern software, then you can leave everything to Raspberry Pi OS (Bookworm)'s default automount behaviour, which will mount the disc in a uniquely named, non-permanent folder under **/media/YourUserName/**. This will happen automatically every time you insert a new disc. You can even use these automatically created mount paths to then mount the disc inside emulators such as DOSBox, as long as you don't need to swap discs.

03 Disc mounting for emulation

Some emulators, but most notably PCSXR and some of the DOSBox family, can only handle disc swapping if all the discs have the same path, and that's not something modern automount behaviours were designed for. If you're interested in emulating games from the late CD-ROM era,


Main
CPU
Video
Sound
DOS
Capture
Drive
Debug
Help
Turbo (Fast Forward)
Normal speed F12+Left
Speed up F12+]
Slow down F12+[
Increment cycles F12+=
Decrement cycles F12+-
Emulate CPU speed
Edit cycles
CPU core
CPU type
8088 XT 4.77MHz (~240 cycles)
286 8MHz (~750 cycles)
286 12MHz (~1510 cycles)
286 25MHz (~3300 cycles)
386DX 25MHz (~4595 cycles)
386DX 33MHz (~6075 cycles)
486DX 33MHz (~12010 cycles)
486DX2 66MHz (~23880 cycles)
486DX4 100MHz (~33445 cycles)
486DX5 133MHz (~47810 cycles)
Pentium 60MHz (~31545 cycles)
Pentium 66MHz (~35620 cycles)
Pentium 75MHz (~43500 cycles)
Pentium 90MHz (~52000 cycles)
Pentium 100MHz (~60000 cycles)
Pentium 120MHz (~74000 cycles)
Pentium 133MHz (~80000 cycles)
Pentium 166MHz MMX (~97240 cycles)
AMD K6 166MHz (~110000 cycles)
AMD K6 200MHz (~130000 cycles)
AMD K6-2 300MHz (~193000 cycles)
AMD Athlon 600MHz (~306000 cycles)
Pentium III 866MHz EB (~407000 cycles)

▲ DOSBox-X has a GUI that makes it easy to tweak your emulation settings on the fly for optimal performance



You'll need to configure appropriate settings, but with Raspberry Pi 5, we were able to play '90s FMV games like *The 7th Guest* without reducing our desktop resolution

During the first Full Motion Video (FMV) era, large chunks of the games were stored on the disc, so you'd have to swap them every now and then while playing

every disc in a set being assigned a unique mount point can be a problem.

It's not always an issue. The DOS emulator, DOSBox-X, has some elegant GUI solutions that make it easier to mount different paths as the same mounted drive. Alternatively, you could create a static mount point and mount each disc as the superuser.

However, we recommend using `pmount` for a hassle-free life. This is a wrapper for the `mount` command that lets any user in the plugdev group create temporary mount points under **/media** for designated devices without elevated sudo privileges or an `fstab` entry to go with them.

04 Mounting discs with pmount

Let's install `pmount` and mount our first disc. Open a terminal and type:

```
$ sudo apt install pmount
$ sudo nano /etc/pmount.allow
```

Add the following to the bottom of the file

```
/dev/sr0
```

Press **CTRL+X**, save and reboot.

Now let's prevent Raspberry Pi from auto-mounting external media. Open the file manager, go to Edit > Preferences > Volume Management and untick all the Auto-mount options, then reboot Raspberry Pi.

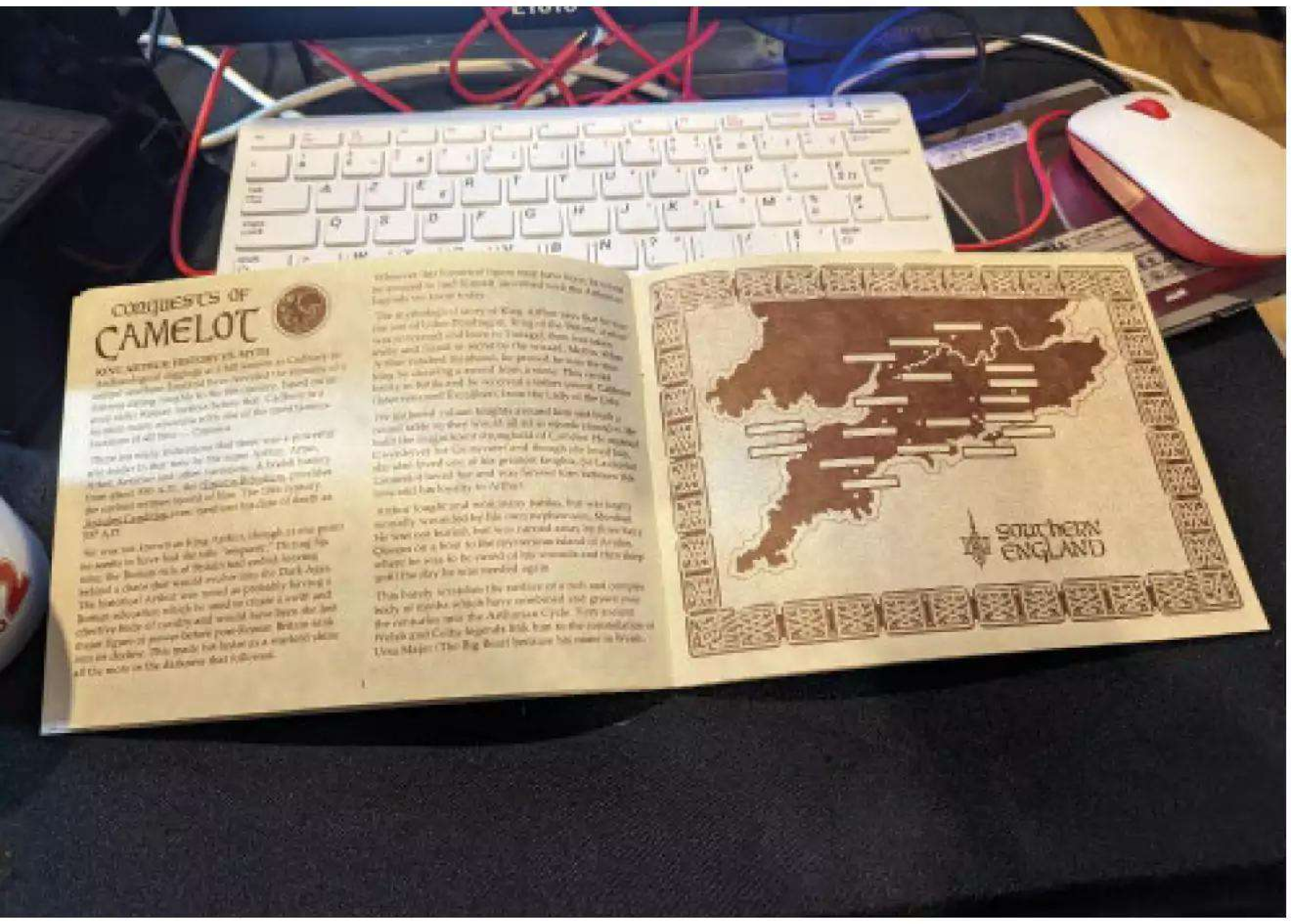
You don't absolutely have to disable automounting, but you'll otherwise have to use `pumount` to unmount an inserted disc before you can assign it to your mount point of choice.

Assuming you've disabled auto-mounting, insert your disc, open a terminal and type:

Warning! Copyright!

This feature discusses making backup copies of floppy disks. Please ensure that you have a licence to reproduce a disk before doing so. Note: The UK government removed the private copying exception in 2015. Archival of software is legal in the United States under section 117 of the Copyright Act. Check the terms and conditions of your software.

copyright.gov
magpi.cc/copyright



Old school DOS games like *Conquests of Camelot* could take a long time to install from floppy discs, but they also came with richly detailed manuals that you could use the time to read

```
$ pmount /dev/sr0 /media/cdrom
```

And to unmount it:

```
$ pumount /dev/sr0
```

05 DOSBox-X and multi-CD gaming

To illustrate disc-swapping, we've used 1993's CD-ROM flagship game, *The 7th Guest*, and the DOSBox-X emulator, which has a GUI that makes the process easier. You can install DOSBox-X as a Flatpak (**magpi.cc/dosboxFH**) or from source. We'll do the latter.

Open a new terminal and type:

```
$ git clone https://github.com/
joncampbell123/dosbox-x.git
$ cd dosbox-x
$ sudo apt install libtool autogen autoconf
automake libncurses-dev gcc g++ make
libncurses-dev nasm libsdl-net1.2-dev
libsdl2-net-dev libpcap-dev libslirp-dev
fluidsynth libfluidsynth-dev libavformat-dev
libavcodec-dev libavcodec-extra libswscale-dev
libfreetype-dev libxkbfile-dev libxrandr-dev
$ ./build-debug
$ make
$ sudo make install
```

06 Disc swapping in DOSBox-X

Now open a second terminal and insert your installation disc (CD 2 for our copy of *The 7th Guest*). Type:

```
$ pmount /dev/sr0 /media/cdrom
```

...return to your first terminal window and type:

```
$ cd
$ mkdir dos
$ dosbox-x
```

While you'd normally use dedicated `conf` files to manage DOSBox-X's behaviour, you can also configure everything on the fly. For *The 7th Guest*, you will want to at least enable OpenGL rendering in the Video > Output menu and switch to an emulated 60MHz Pentium in the CPU > Emulate CPU settings. Then, in DOSBox-X's main window, type the following:

```
$ mount C ~/dos
$ mount D /media/cdrom -t cdrom -usecd 0
-ioctl
$ D:
$ install
```

Whenever the game prompts you to insert another CD, it's best to switch to the other terminal window and type:

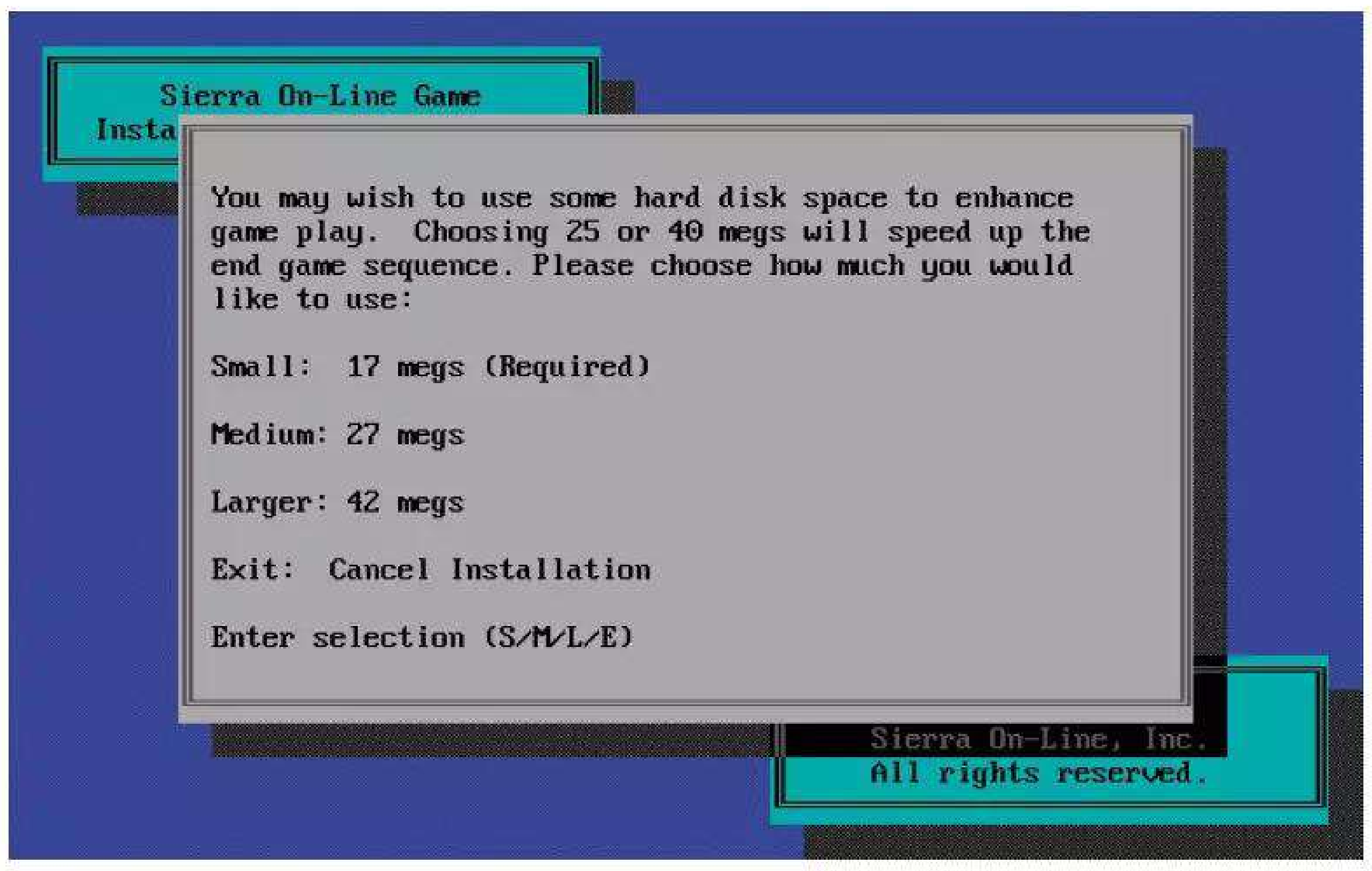
```
$ pumount /dev/sr0
```

Eject the drive, insert the new disc, then type:

```
$ pmount /dev/sr0 /media/cdrom
```

You can also get away with manually ejecting the disk and simply repeating the `pmount` command.

After that, in DOSBox-X, tell it that you've swapped discs via its GUI by going to Drive > D > Rescan drive, click to tell the game to read the new disc, and carry on playing.



```
Sierra On-Line Game
Insta

You may wish to use some hard disk space to enhance
game play.  Choosing 25 or 40 megs will speed up the
end game sequence. Please choose how much you would
like to use:

Small:  17 megs (Required)

Medium: 27 megs

Larger: 42 megs

Exit:  Cancel Installation

Enter selection (S/M/L/E)

Sierra On-Line, Inc.
All rights reserved.
```

▲ Early CD-era games would only occupy a relatively small space on your hard disk. Gabriel Knight 2 came on six CDs, but occupied a maximum of 42 MB on your hard disk

07 Reading floppies

The same `pmount` method works with floppy disks. You'll need an external reader that you can connect via USB. We used a Dell MPF82E, a swappable laptop floppy drive that you can also connect via Mini USB. There are quite a few of these around, and they tend to be pretty tough as far as external drives go. Alternatively, you could use an internal floppy to USB adaptor such as a Greaseweazle (**magpi.cc/greaseweazle**). You'll need to provide your own vintage drive to use it, but an adaptor of this sort is your only option if you need to read 5.25in floppies on a modern computer.

08 Mounting floppies

The easiest way to set up emulator-agnostic floppy disk swapping is the same method we used for CDs earlier. Like CDs, floppy disks, as are by default mounted with unique paths, which can interfere with disk-swapping in some emulators, so we'll again use `pmount` and `pumount`.

USB floppy drives consistently appear in Raspberry Pi OS (Bookworm) under **/dev/sda**, but you can use the `lsblk` command at the terminal to list all block devices and double-check how you should be addressing them.

Add the following to the **pmount.allow** file we edited earlier:

```
/dev/sda
```

You can now mount and unmount floppy disks thus:

```
$ pmount /dev/sda /media/floppy
$ pumount /dev/sda
```

> " An adaptor is your only option if you need to read 5.25in floppies "

Once again, you'll probably want to ensure that auto-mounting is disabled in the file manager unless you want to have to manually `pumount` each floppy.

09 Multi-floppy installs in DOSBox

Launch DOSBox-X from either a fresh terminal or your window manager. Then, in DOSBox-X, type:

```
$ mount  c ~/dos
$ mount a /media/floppy -t floppy
$ a:
$ install
```

Wait what might feel like a shockingly long time for the disk to unpack its contents. Read the manual. Make some tea. Installing from floppies was never quick, and installing from 30-year-old floppies is even slower. When you're finally told to insert disk 2, if you're using DOSBox-X and a `pmount` point as discussed, you can just physically swap the disks in your floppy drive and press the specified key to continue, without any need to unmount or remount your floppy drive for each disk. Repeat this process, and carry on until you run out of floppies.

10 Can you copy that floppy?

Many – but by no means all – floppy disk software licences included explicit permission to copy the disks or their contents to other media for strictly personal use. Many installers simply extract and/or copy the full game files to their installation target.

You'll find these licences apply not only to obviously copyable content such as free software, shareware applications and game demo disks, but sometimes even operating systems such as MS-DOS 5 and games such as *Ultima V*.

However, if the documentation and licence agreements that accompany a commercial floppy disk do not allow you to copy it, it is safest to assume that you are not allowed to do so.

Those that do not allow you to make copies often include copy protection that will prevent you from

Top Tip

Shutter island

You may need to gently but firmly slide your floppy disk's metal shutter back and forth before it'll read. Do not bend this!

Top Tip

Floppy rescue

If you're having serious trouble with recovering the contents of a floppy disk, you should also try the Ddrescue recovery tool (**magpi.cc/ddrescue**).

doing so, and it is illegal in the UK to circumvent this. While some disks were made for reproduction, unless there's an obvious licence allowing you to, the old rule holds: don't copy that floppy.

11 Imaging floppies

Because almost all floppy disks have now reached the end of their useful lifespan, you absolutely should be backing up both software and personal files in the form of disk image files and/or file copies. File copies are easy: just mount the disk and copy everything you find in it over to a backup directory. But images are more useful for preserving data in its original state.

We know Raspberry Pi mounts floppies at **/dev/sda**, but let's get more information about the disk. Insert your disk, open a terminal and type:

```
$ sudo fdisk -l
```

In our case, fdisk showed us that we had a high density floppy mounted at **/dev/sda**, packing in a massive 1.44 megabytes of data, so our copy command looks something like:

```
$ sudo dd bs=512 count=2880 if=/dev/sda
of=my_disk_image.img
```

12 Restoring your backups

If you've made backups and actually own floppy disks, you can reverse this process to copy your backup to a new floppy disk, thus:

```
$ sudo dd bs=512 count=2880 if=my_disk_image.
img of=/dev/sda
```

We had to do this to replace a non-functional original floppy during testing, but it's also a good idea to make sure that any backups you make can actually be restored. If you don't have spare floppies lying around, you can mount your floppy disk IMG files as a virtual drive with the following command:

```
$ sudo mount -o loop disk1.img /mnt
```

Then just browse to **/mnt** to examine their contents. Some tools, including DOSBox-X, can also directly mount IMG files.

In contrast to their later CD-ROM counterparts, most floppy disk era games would copy all their files to your hard disk so you could play them whenever (and however) you wanted

RETRO GAMING WITH RASPBERRY PI

3RD EDITION

Retro Gaming with Raspberry Pi shows you how to set up Raspberry Pi 5 to play a new generation of classic games. Build your gaming console and full-size arcade cabinet, install emulation software and download original games with our step-by-step guides. You'll discover a vibrant homebrew scene packed with new games for original consoles and legal access to all those retro games you remember!

- *Set up Raspberry Pi for retro gaming*
- *Emulate classic computers and consoles*
- *Learn to code retro-style games*
- *Build a console, handheld, and full-size arcade machine*

FROM THE MAKERS OF The MagPi THE OFFICIAL RASPBERRY PI MAGAZINE
START
NEW 2024 UPDATE
RETRO GAMING WITH RASPBERRY PI
3RD EDITION
180 PAGES OF VIDEO GAME PROJECTS
BUILD AN ARCADE MACHINE

BUY ONLINE: **magpi.cc/store**

Part 02

Upcycle a Sonos Play:1

So far, we have converted a Sonos Play:1 into a Raspberry Pi-driven powerhouse of sound. Now it's time to trick it out with software

MAKER

PJ Evans

PJ is a writer, software engineer and tinkerer. He got fed up listening to Rick Astley (bit of a let down) and now plays The Stick Song on repeat.

mrpjevans.com

You'll Need

- Raspberry Pi Smart Speaker build **magpi.cc/139**
- Installation of Home Assistant (optional)

Whether you're considering converting your Sonos Play:1 or perhaps upcycling a vintage radio, you should have a project in mind by now that'll fill your room with sound, Raspberry Pi-style. Previously, we walked through the hardware build of our super-smart speaker project but so far we have only been able to play 'white noise'; where are the tunes? Now that we know the audio is working, we can focus on what we want to listen to and how to control it. We've selected some common uses for smart speakers, but feel free to explore your own ideas!

01 Wired for sound

We start off with a small stereo problem. The Sonos Play:1 only has a single speaker and it's not a great idea to wire it to both outputs of the amp HAT. Instead, we can reconfigure sound output to mix both channels to the left speaker (which is the one that should be wired up). We can prove this by running:

```
speaker-test -c 2
```

The Play:1 will only play the left-hand white noise signal (**CTRL+C** to stop). To fix this, start by getting the 'number' of your card by running the following command:

```
$ aplay -l
```

The output will probably either start with 'card 0' or 'card 1'. Make a note of it.

02 Two become one

Luckily, we can fix our stereo problem by mixing the two channels in software. To do this, we'll configure the ALSA audio sub-system. Start by creating a global configuration file (it may already exist):

```
$ sudo nano /etc/asound.conf
```

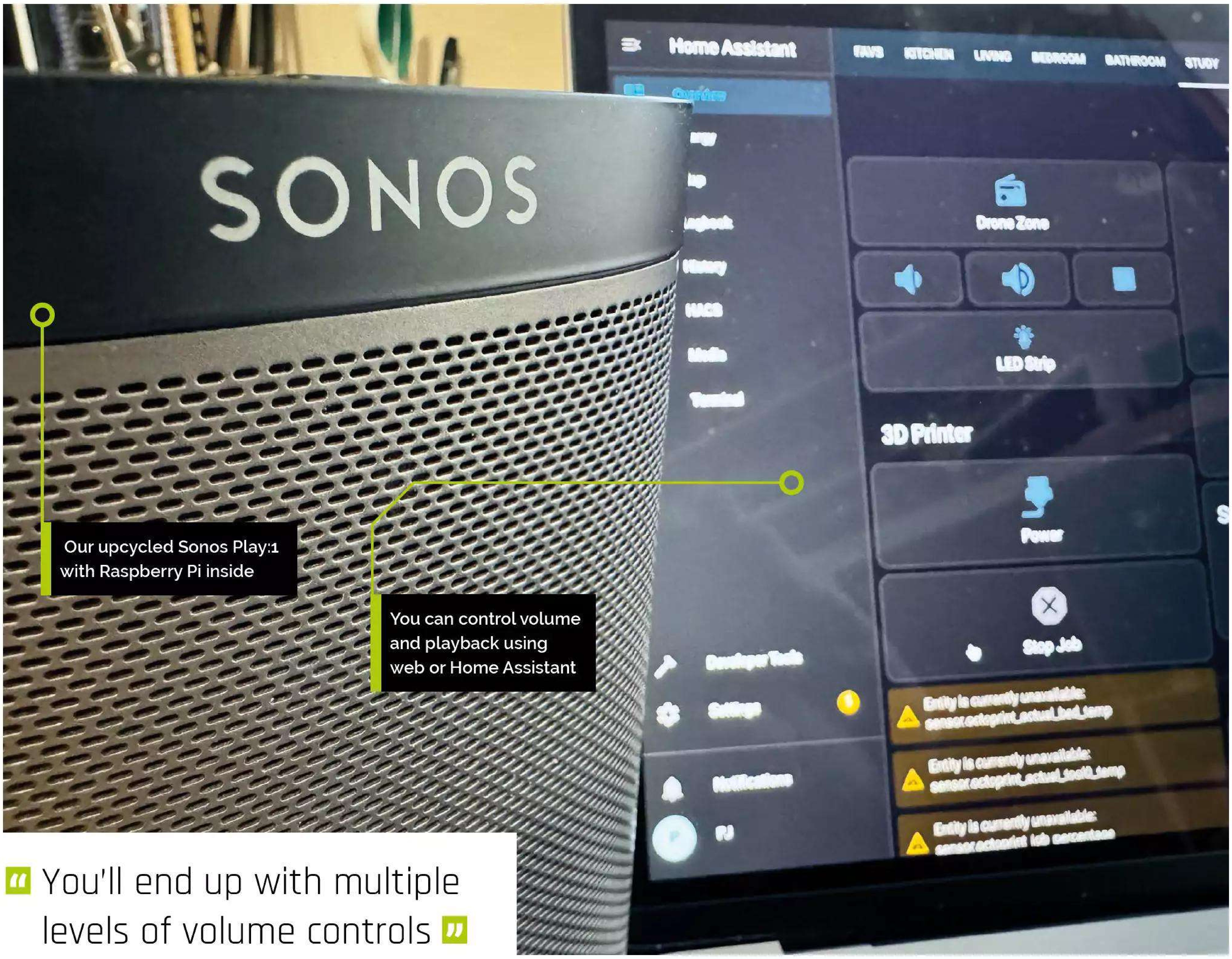
If it already exists, replace all the text with the contents of the code listing. You can also grab the code from **magpi.cc/asoundconf**. If your card number was something other than 0, then make sure you replace all instances of 'card 0' with the correct number (e.g. 'card 1'). Save the file (**CTRL+X** followed by **Y**). No need to reboot, just run `speaker-test -c 2` again and now both channels should be played from the speaker.

03 Pump up the volume

Before tricking out our new speaker, it's a good idea to set the base volume. You'll end up with multiple levels of volume controls depending on your choices. For example, streaming from a phone will have the phone's output volume feeding to Raspberry Pi's volume. If you find you want to set the output of Raspberry Pi's amp, just run the `alsamixer` from the terminal. You'll see a rudimentary volume control where you can use the cursor keys to adjust the master volume output. If you can't see any controls, press **F6** to get a list of available sound devices. When done, **ESC** will return you to the command line.

04 Local hero

Maybe at this point, you just want to be able to play some music or other audio. After all, there's no need to turn this into a


SONOS
Home Assistant
FAVS
KITCHEN
LIVING
BEDROOM
BATHROOM
STUDY
Drone Zone
LED Strip
3D Printer
Power
Stop Job
Developer Tools
Settings
Notifications
PJ
Entity is currently unavailable: sensor.octoprint_actual_bed_temp
Entity is currently unavailable: sensor.octoprint_actual_tool0_temp
Entity is currently unavailable: sensor.octoprint_job_percentage

Our upcycled Sonos Play:1 with Raspberry Pi inside

You can control volume and playback using web or Home Assistant

> You'll end up with multiple levels of volume controls

asound.conf

> Language: **BASH**

DOWNLOAD THE FULL CODE:

magpi.cc/asoundconf

```
pcm.card0 {
  type hw
  card 0
}

ctl.card0 {
  type hw
  card 0
}

pcm.monocard {
  slave.pcm card0
  slave.channels 2
  type route
  ttable {
    # Copy both input channels to output channel 0 (Left).
    0.0 0.5
    1.0 0.5
    # Send nothing to output channel 1 (Right).
    0.1 0
    1.1 0
  }
}

ctl.monocard {
  type hw
  card 0
}

pcm.!default monocard
ctl.!default monocard
```

streaming device, you've already got enough of the project working to play audio. If you've installed a desktop environment, you could also use it as your computer! For those comfortable with the command line, you can upload your audio files and play them back with mpg321, a handy command line utility. To install: `sudo apt install mpg321`. There are many apps to help you manage and listen to your music collection such as **volumio.com**.

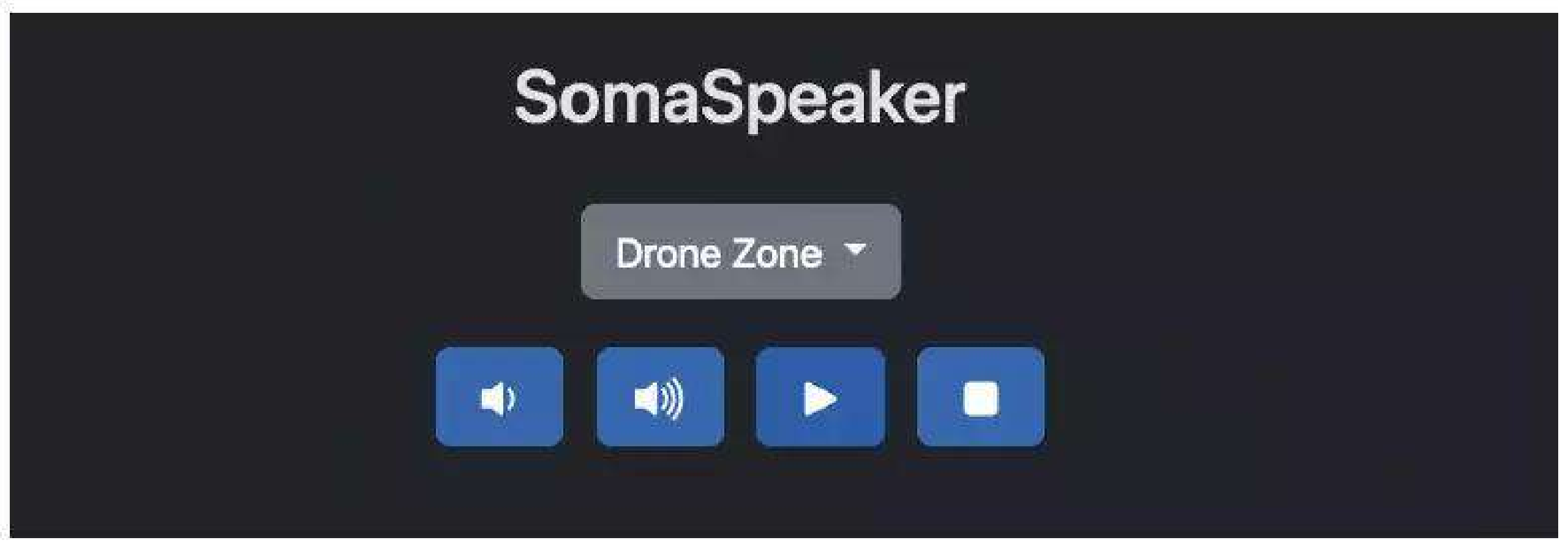
05 Story of the blue(tooth)s

At this point, it would be reasonable to expect a section on converting our speaker into a Bluetooth-capable endpoint. Sadly, Bluetooth is a complex beast and although it is possible to turn the build into a Bluetooth speaker, there are so many different approaches that vary depending on which hardware or operating system you are running, that it would take several tutorials to cover all eventualities. So, to summarise, it is possible, but requires a lot of low-level hacking about. If you still want to give it a try, start with this guide: **magpi.cc/bluealsa**.

06 In the air(play) tonight

Although Bluetooth is a bit of a pain, the good news is that owners of Apple devices can create the same capability using the AirPlay standard. In the terminal use the following commands to build and install the latest version (which supports Raspberry Pi 5):

```
$ sudo apt install autoconf libtool libdaemon-
dev libasound2-dev libpopt-dev libconfig-dev
libssl-dev libavahi-client-dev git
$ cd
$ git clone https://github.com/mikebrady/
shairport-sync.git
$ cd shairport-sync/
$ autoreconf -i -f
$ ./configure --with-alsa --with-avahi --with-
ssl=openssl --with-systemd --with-metadata
$ make
$ sudo make install
$ sudo systemctl enable shairport-sync
$ sudo systemctl start shairport-sync
```



Our basic web interface is a good starting point for your own project

Once installed and running, your iPhone/iPad/Mac should be able to see 'Audio' as a streaming target. Try it out and you should get sweet, sweet music from your speaker. You can change the name by editing the config file: `sudo nano /usr/local/etc/shairpoint-sync.conf`. You'll need to reboot after this.

07 Radio waves

We'll now turn our attention to media streaming. There are countless audio streaming services out there and one of our favourites is SomaFM, a collection of ad-free eclectic stations providing music for all kinds of tastes. We like its ambient channels which are both relaxing and can aid concentration. In the next few steps, we're going to turn our speaker into a SomaFM receiver for Drone Zone, SomaFM's most popular ambient service. Then we'll add MQTT support so we can control the speaker remotely. There's a lot of code involved here, so we have prepared a repo with everything you need. From the terminal:

```
$ cd
$ sudo apt install jq mpv python3-pip
$ sudo pip3 install flask paho-mqtt
$ git clone https://github.com/mrpjevans/
somaspeaker.git
```

08 Station to station

Let's make sure you can listen to SomaFM. Try these commands:

```
$ cd ~/somaspeaker/
$ ./somafm.sh channels
```

…you should have a list of channels to choose from. Now try this:

```
$ ./somafm.sh listen dronezone
```

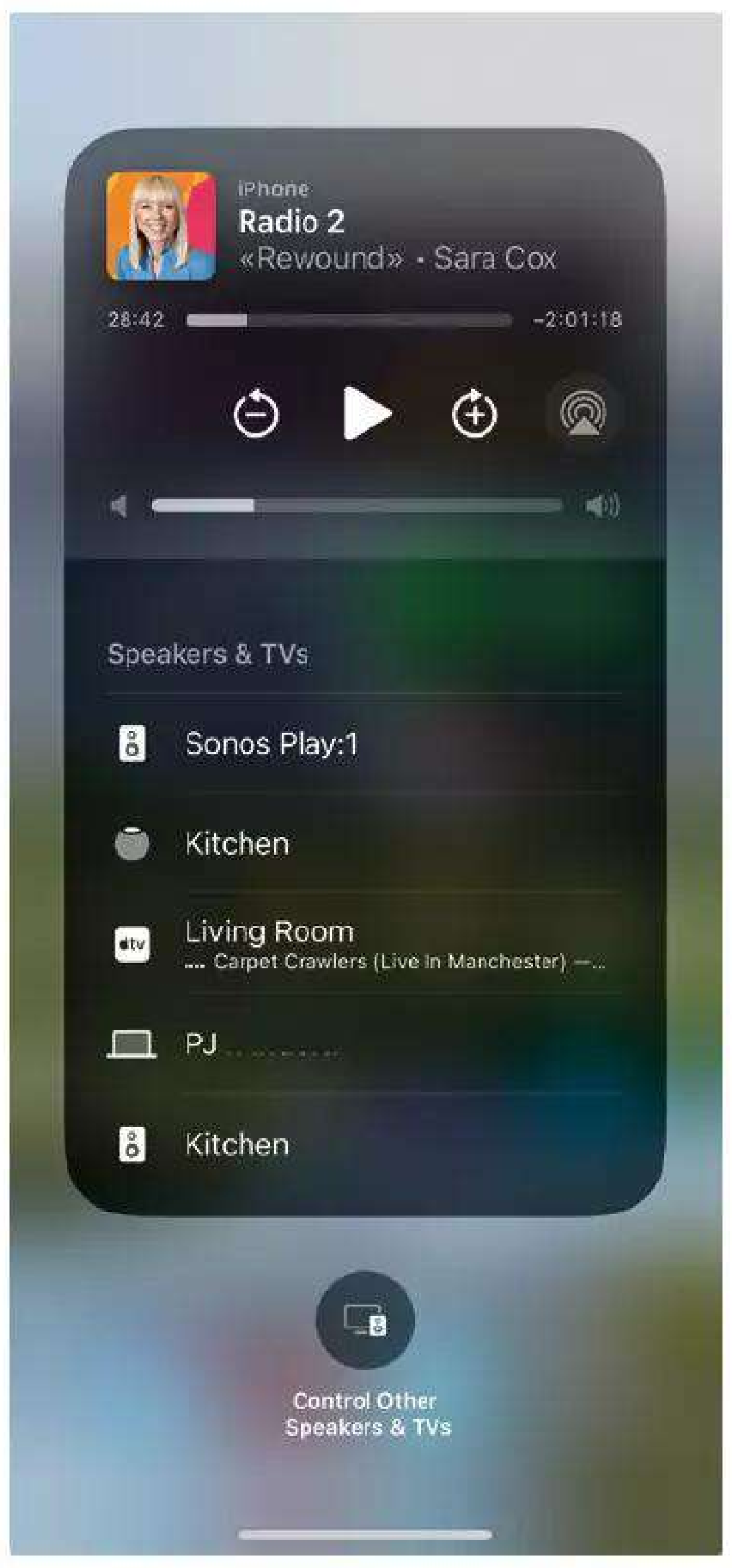
If you can't hear much, check alsamixer for the volume. Use **CTRL+C** to stop when you're done. Have a listen to the other channels too, you might find a favourite!

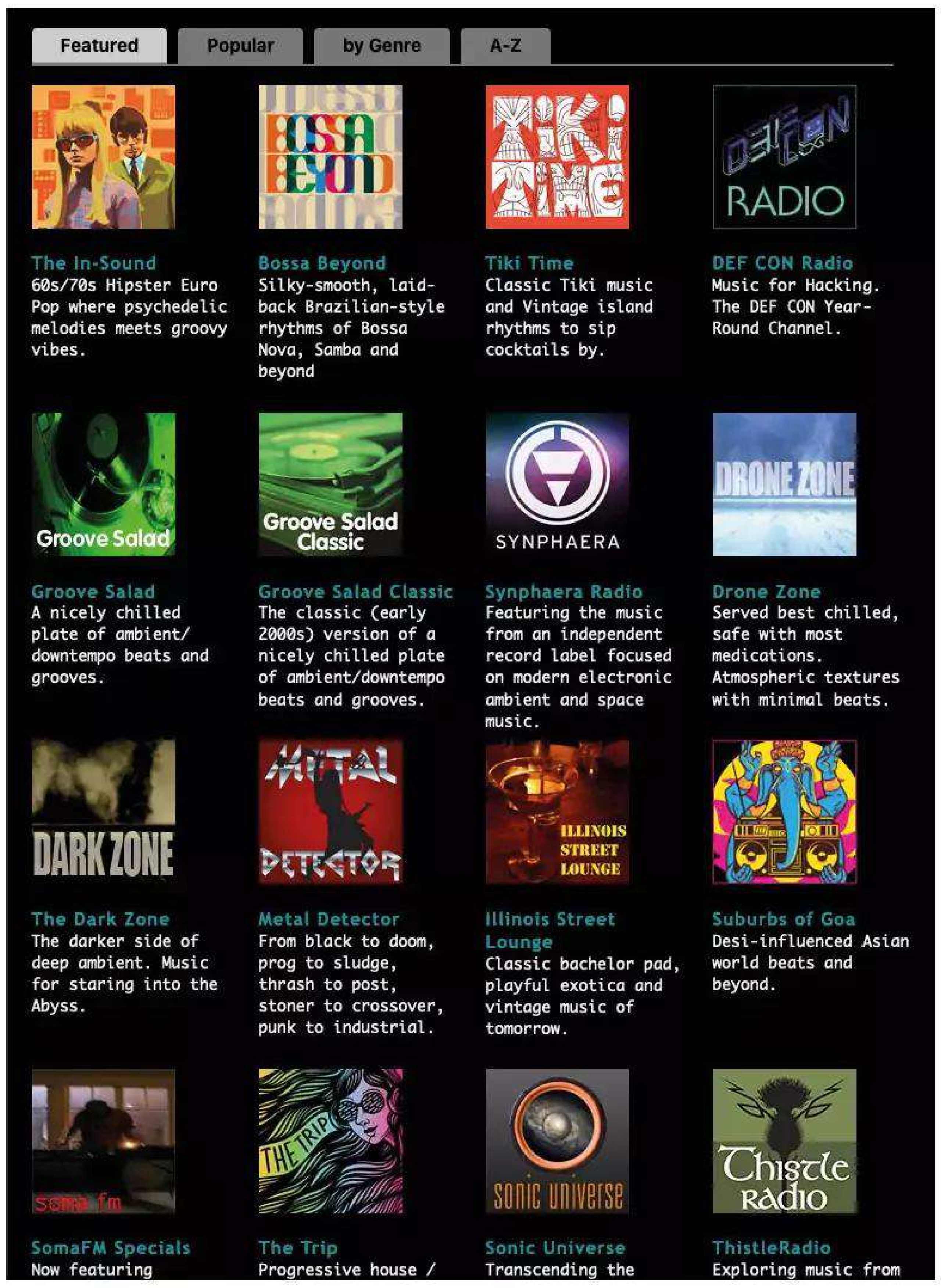
Top Tip

Support SomaFM

SomaFM is free to listen and, bar the odd ident, completely ad-free. If you build this project, consider supporting them: **somafm.com**.

Here's how the AirPlay service appears on your Apple device





The In-Sound
60s/70s Hipster Euro Pop where psychedelic melodies meets groovy vibes.

Bossa Beyond
Silky-smooth, laid-back Brazilian-style rhythms of Bossa Nova, Samba and beyond

Tiki Time
Classic Tiki music and Vintage island rhythms to sip cocktails by.

DEF CON Radio
Music for Hacking. The DEF CON Year-Round Channel.

Groove Salad
A nicely chilled plate of ambient/downtempo beats and grooves.

Groove Salad Classic
The classic (early 2000s) version of a nicely chilled plate of ambient/downtempo beats and grooves.

Synphaera Radio
Featuring the music from an independent record label focused on modern electronic ambient and space music.

Drone Zone
Served best chilled, safe with most medications. Atmospheric textures with minimal beats.

The Dark Zone
The darker side of deep ambient. Music for staring into the Abyss.

Metal Detector
From black to doom, prog to sludge, thrash to post, stoner to crossover, punk to industrial.

Illinois Street Lounge
Classic bachelor pad, playful exotica and vintage music of tomorrow.

Suburbs of Goa
Desi-influenced Asian world beats and beyond.

SomaFM Specials
Now featuring

The Trip
Progressive house /

Sonic Universe
Transcending the

ThistleRadio
Exploring music from

SomaFM is a great collection of music genres

09 Ready to go

Now we have our SomaFM streaming device, we could use a way of being able to remote control both playback and volume control. For this project, we're going to create a small web service which can be accessed on your local network. The code is already in the repo you downloaded in the previous step and we encourage you to investigate it and make changes to suit. To start the service manually, just run this from the directory:

> "Listen to the channels, you might find a favourite!"

```
$ flask --app somaspeaker run -h 0.0.0.0 -p 3000
```

You should now be able to access the site on http://(your IP address or hostname):3000/. Try starting the station and changing the volume.

10 Start it up

Let's round things off by making sure our web site starts when the Raspberry Pi boots. To do this we need a service file. To save you some typing, we've got one ready to go. Make sure the web server is not running. Now, run the following commands to set everything up:

```
$ cd ~/somaspeaker
$ sudo ./install_web_service.sh
```

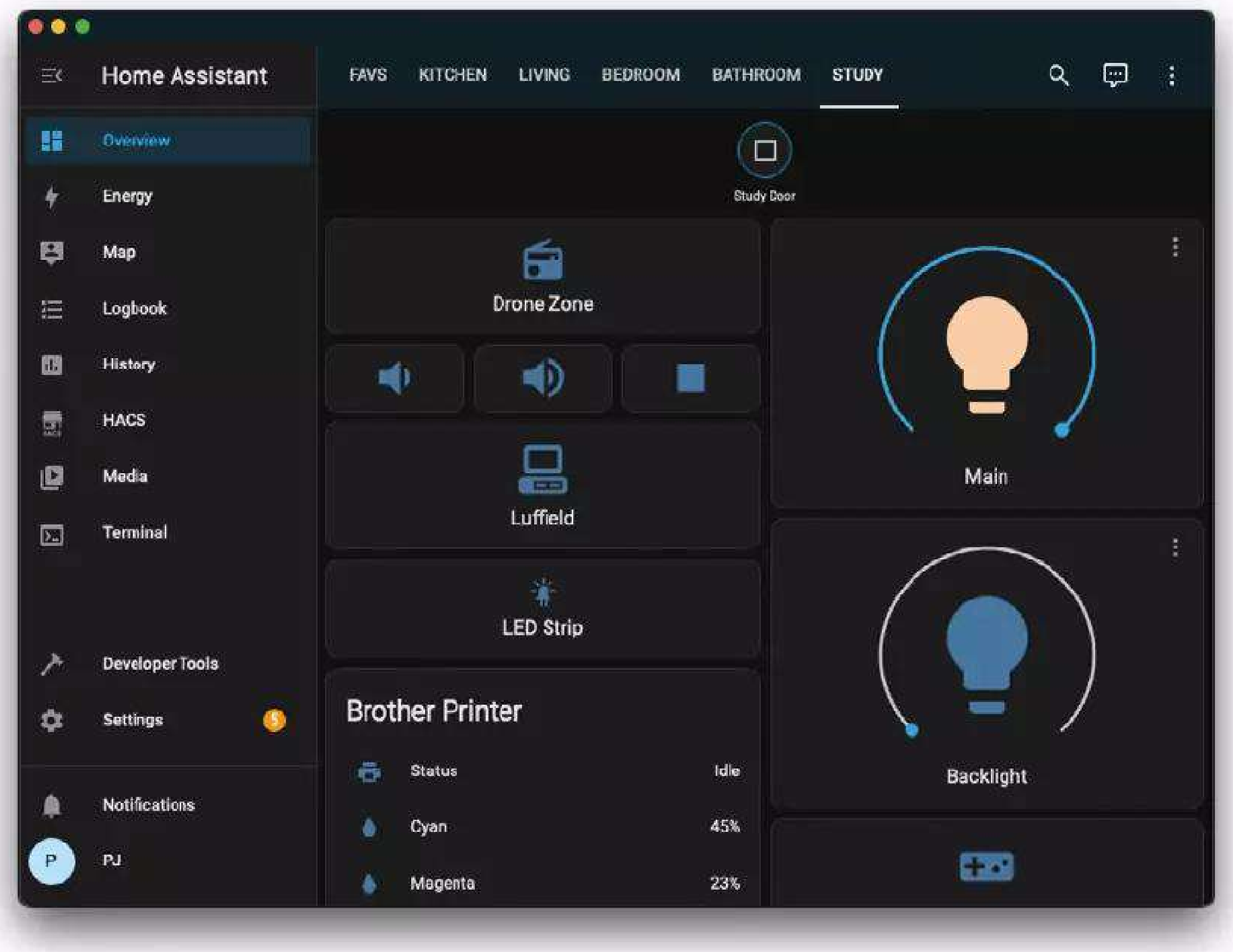
You should be able to access the server as before. Reboot your Raspberry Pi and you should find the server is automatically running.

11 Message in a bottle

If you are using Home Assistant, you can embed controls for your speaker right into the dashboard. To do this you'll need an MQTT broker such as Mosquitto. If you have one, edit the `config.py` file in `~/somaspeaker` to enable MQTT support and set the broker's address. Upon restart the server will subscribe to the given topic and listen for instructions. You can then use Home Assistant's MQTT integration to send commands to the speaker. See the provided README.md for examples.

12 More, more, more

You've now got a smarty-pants speaker that can play local files, act as an AirPlay target and stream SomaFM channels. That doesn't have to be everything. You can try adding other streaming services, or create a cool Jukebox app. How about AI-generated playlists, or generative music which can react to light and temperature? Take Home Assistant further by automating playback (e.g. when you switch a light on). Whatever you decide, you have hopefully prevented another piece of e-waste from going into landfill and you've got a great speaker system to go with it.



Brother Printer

Status Idle
Cyan 45%
Magenta 23%

Top Tip

Equal rights

If you would like more control over sound quality, equaliser plugins are available for ALSA. Example: **magpi.cc/alsaequal**

You can customise Home Assistant control playback, even adding automations

Pico-powered Hull Pixelbot

Make a Pico-powered pixel packing robot you can program from your browser in 'Python-ish'

Rob Miles

Rob Miles has been playing with hardware and software since almost before there was hardware and software. You can find out more about his so-called life at **robmiles.com**.

Find out how to create a little robot you can program from your PC browser. Along the way, we'll dig into the deepest recesses of computer interrupts, build a programming language that has angry and happy commands, and use a web-based robot development environment.

The Hull Pixelbot is a little robot with a pixel on the top. **Figure 1** shows the very first one. The author wanted to find out just how much you could do with a robot powered by an Arduino Uno, a very popular (if somewhat limited) microcontroller. Version 1 just moved around the desk. Version 2 had a pixel and distance sensor and became the Hull Pixelbot. At first the robot was controlled by C++ compiled on a PC and loaded into the robot. But then a tiny programming language, 'Python-ish', was developed that runs directly on the robot. The robot can obey Python-ish commands directly at any time – even when running a program. Compiled code can be stored on the robot and executed automatically when the robot is switched on. You can add a network connection (coming next month) so that you can update the program from anywhere.

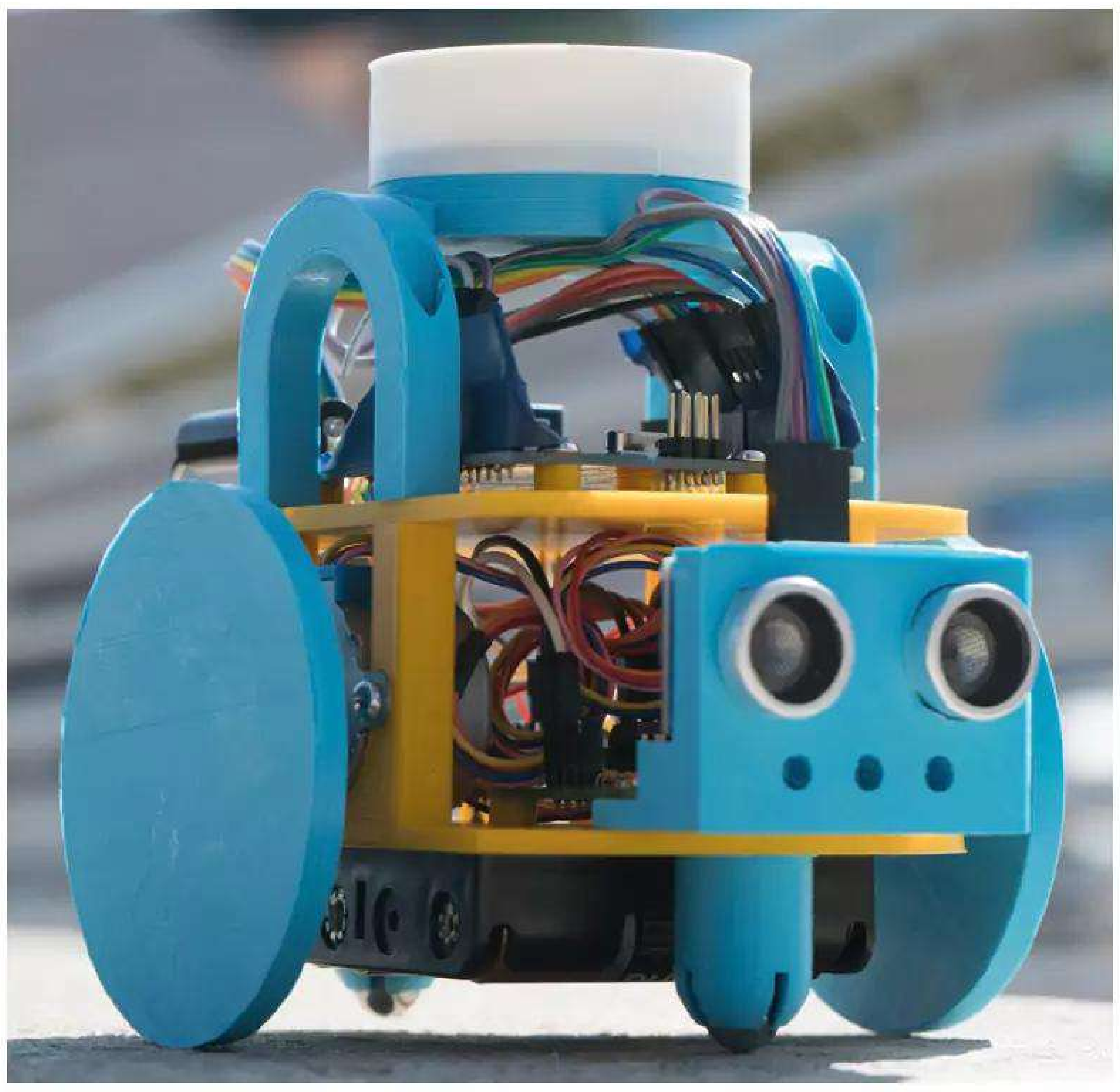


Figure 1 It's called the Hull Pixelbot because it was created in the city of Hull in the UK

The Pixelbot is a great way to upcycle old Arduinos and use them to power fun robots. There are also versions powered by the Raspberry Pi Pico or the Espressif ESP32. You can download the software and find out all about the project at **hullpixelbot.com**.

ROBOT CIRCUITRY

Figure 2 shows the circuit diagram for the robot. The devices fit into a 400-point breadboard which sits on top of the robot. There are plenty of spare connections for other interfaces you might want to add. Let's start by looking at the various devices on the robot, starting with the one that tells the robot if there is something in front of it.

MEASURING DISTANCE

Robots that crash into things look silly. So, the robot has a distance sensor on the front to detect any wayward brick walls. The HC-SR04 sensor sends out a burst of high-frequency sound and generates a signal pulse when it gets a response. Software in the robot measures the length of the pulse and because we know the speed of sound, it can calculate the distance the robot is from an obstacle. The distance sensor uses two signals. The trigger signal asks the sensor to start a measurement and send a burst of sound. The sensor lifts the echo signal to 5 volts when the sound reflection is detected. We could write a function to trigger the sensor and spin in a loop counting and waiting for the echo signal to go up. This would work, but it would mean that the program would have to stop while distance reading is taken. Sound travels at around 330 metres a second, so if an obstacle is a metre away, it would take around 150th of a second for the reading to be taken. This doesn't sound like much, but the author hates writing code that waits around for a response, so hardware interrupts are used to process the echo signal.

MAKING A DEVICE COME ALIVE

The author was very keen to make the robot seem a little alive. Most robots tend to do only one thing at once – a Hull Pixelbot can animate its pixel, move, and measure distance, all at the same time. Programs can even express emotion. The 'angry' command will make the lights in the pixel flash on and off aggressively; the 'happy' command makes them fade on and off in a much more relaxed fashion.

IF I MAY INTERRUPT YOU…

The first computers didn't have interrupts. Programs had to constantly check for changes to their inputs. The interrupt was added in the 1950s. Hardware in the computer processor detects a change in state of an input and diverts program execution to a piece of code called an 'interrupt handler' to process the incoming event. When the event has been dealt with, the original program continues. Interrupts are part of how modern computers work. Whenever you press a key, move the mouse, or your computer receives a network message, an interrupt handler takes control to deal with the event. We can use interrupts to allow the robot control program to run while a distance reading is being taken.

> The 'angry' command will make the lights in **the pixel flash on and off aggressively**

```
#define DISTANCE_TRIG_PIN 17
#define DISTANCE_ECHO_PIN 16
```

The code above defines the two symbols which will represent the trigger and echo pin numbers to be used on the Pico. Using symbols makes it easy for us to change the pin numbers if the hardware design changes. We are going to use pins 16 and 17.

```
pinMode(DISTANCE_TRIG_PIN, OUTPUT);
pinMode(DISTANCE_ECHO_PIN, INPUT);
```

The two statements above set up the pins. The trigger will be an output, and the echo an input.

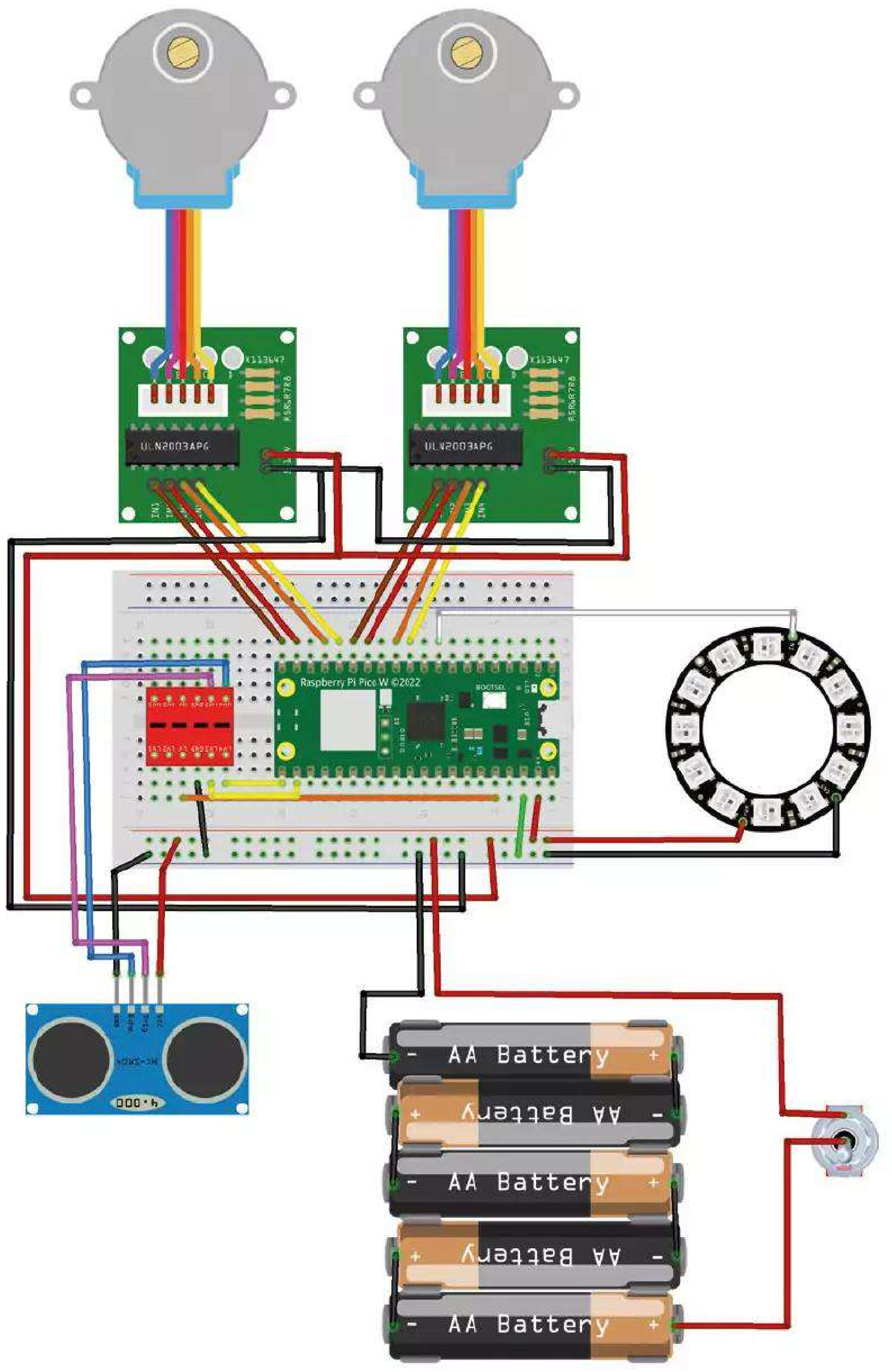


Figure 2 The red component is a level converter which is needed because the distance sensor uses 5-volt signals rather than the 3.3 volts used by the Pico

```
attachInterrupt(digitalPinToInterrupt(DISTANCE_ECHO_PIN),
    pulseEvent,
    CHANGE);
```

This statement attaches an interrupt handler to the echo pin. The **digitalPinToInterrupt** function takes the number of the interrupt pin and converts it into a corresponding number for the specific hardware in use. The handler function is called **pulseEvent** and the interrupt will fire when the pin changes state (either goes from high to low or low to high).

```
void pulseEvent()
{
    if(digitalRead(DISTANCE_ECHO_PIN)) {
    // pulse gone high - record start
    pulseStartTime = micros();
  }
  else
  {
    pulseWidth = micros() - pulseStartTime;
    distanceSensorState = DISTANCE_SENSOR_READING_READY;
  }
}
```

→

YOU'LL NEED

- **A Raspberry Pi Pico or Pico W.** If you want to control the robot over WiFi, use the Pico W version. If you have an old Arduino Uno, you can use that instead, although you'll have to add an ESP8266 to connect your robot to WiFi.
- **A 12-pixel NeoPixel Ring**
- **A 4 * AA battery holder with switch**
- **A couple of 28BYJ-48 stepper motors with ULN2003 Driver Board** Search for 'Arduino stepper'
- **An HC-SR04 distance sensor**
- **20 or so plug-to-socket 20cm DuPont cables**
- **A 400-point solderless breadboard**
- **A level converter chip** (search for 'arduino level shifter' on your favourite e-commerce site)
- **A USB cable**
- **A chassis and some wheels** You can make your own or there are 3D-printable or laser-cuttable designs you can download from hullpixelbot.com

QUICK TIP

Not all microcontrollers provide byte-wide input and output. When using the Pico, we have to set each output bit in turn.

The `pulseEvent` function runs when the echo signal changes state. The echo signal goes high at the start of a reading and low when the response has been received. The code above checks to see if the pulse has gone high. If it has, it records the current time in microseconds. If the pulse has gone low, the code determines the width of the pulse by subtracting the start time from the current time in milliseconds. It then sets a flag to indicate that a new distance sensor reading has been obtained. The robot code can check this flag every now and then to see if a new reading has arrived and update the `@distance` value in the Python-ish program running on the robot. This means you can use constructions like the code below which makes the robot pixel turn red and flash in an angry way if you get too close.

```
if @distance < 100:
    angry
    red
```

SPINNING THE WHEELS

Stepper motors are used to turn the robot wheels. These are slower than servos or simple electric motors, but they have much greater precision. You can instruct the robot to move 100 millimetres and it will move exactly that distance. The author is happy to trade speed for accuracy. He's found that fast-moving robots tend to crash into things or fall off the table.

Figure 3 represents the inside of the robot stepper motor. The round thing in the middle is the motor shaft, and the block on the shaft is a little magnet. If we turn on one of the four coils, the magnet is attracted to that coil. In **Figure 3**, coil 1 is turned on and the magnet has been pulled towards it. We can make the shaft turn by performing the following:

- Turn on coil 2 (magnet is pulled between coils 1 and 2)
- Turn off coil 1 (magnet is pulled to coil 2)
- Turn on coil 3 (magnet is pulled between coils 2 and 3)
- Turn off coil 2 (magnet is pulled to coil 3)
- Turn on coil 4 (magnet is pulled between coils 3 and 4)
- Turn off coil 3 (magnet is pulled to coil 4)
- Turn on coil 1 (magnet is pulled between coils 4 and 1)
- Turn off coil 4 (magnet is pulled to coil 1)

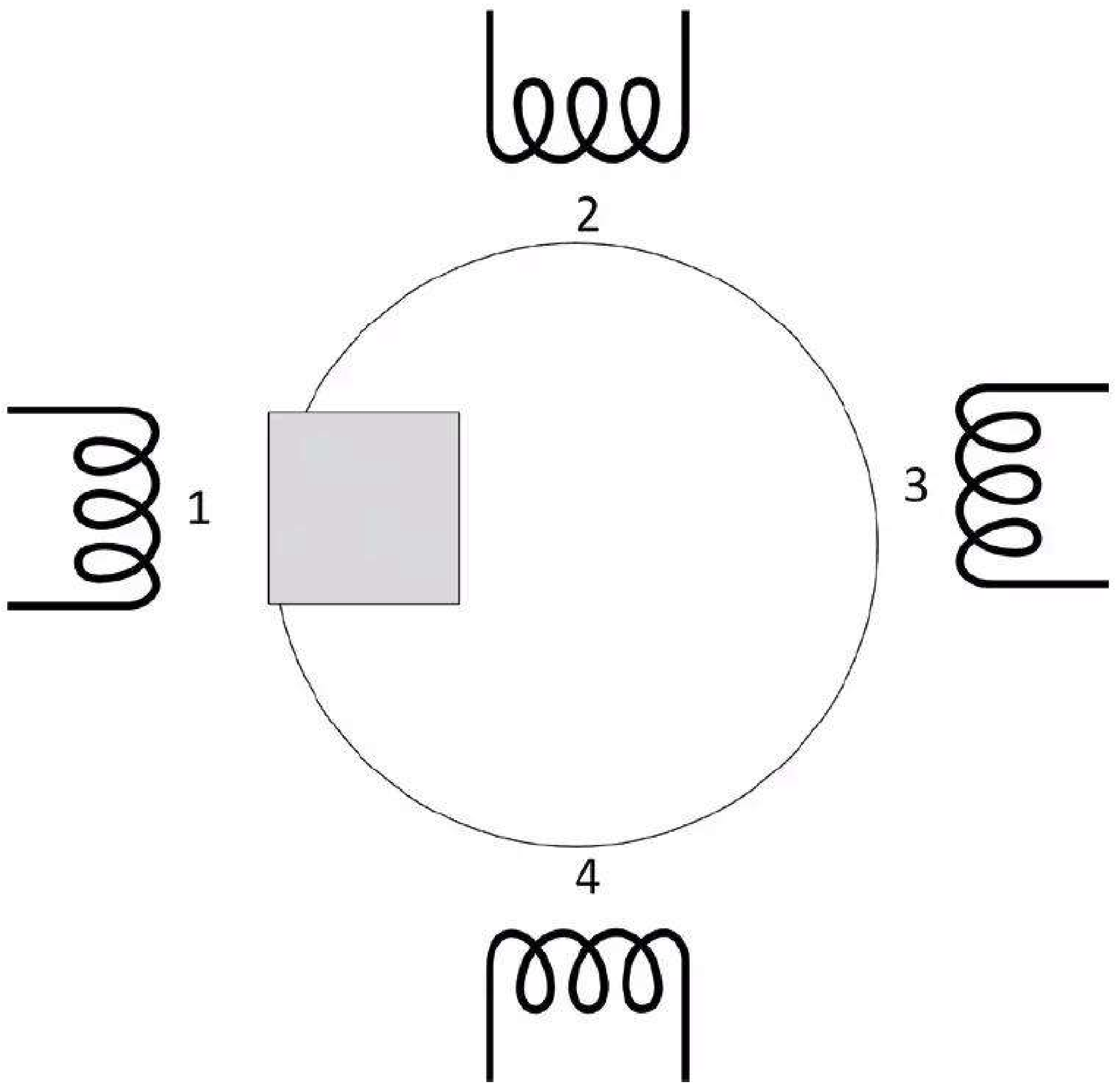


Figure 3 The coils and magnets are arranged slightly differently in a real motor, but the principle is the same

A program that repeated these changes would make the shaft turn continuously. We can represent the motor coil settings using an array of eight bytes:

> "You can instruct the robot to move 100 millimetres **and it will move exactly that distance**"

```
const byte rightMotorWaveformLookup[8] = { 0b01000,
0b01100, 0b00100, 0bB00110, 0b00010, 0b00011,
0b00001, 0b01001 };
```

If you look at the bit patterns in the binary constants in the code above, you can see that coil 1 is represented by bit `0b01000`, coil 2 by bit `0b00100`, and so on. In the Arduino Uno, these bit patterns can be loaded directly into an output register to turn on the specified outputs.

```
PORTB = (PORTB & 0xF0) +
   rightMotorWaveformLookup [rightMotorWaveformPos];
```

The statement above takes a value from the lookup table and drops it onto the bottom four bits of the `PORTB` output register, setting all four signal bits at the same time. This is much more efficient than setting each output bit individually.

A program can make the right-hand motor turn just by stepping `rightMotorWaveformPos` from 0 to 7, which would access each of the elements in the `rightMotorWaveformLookup` array. The program would need to pause between updates to allow the magnet to move into position. We can control the speed of the motor by changing the delay between each update.

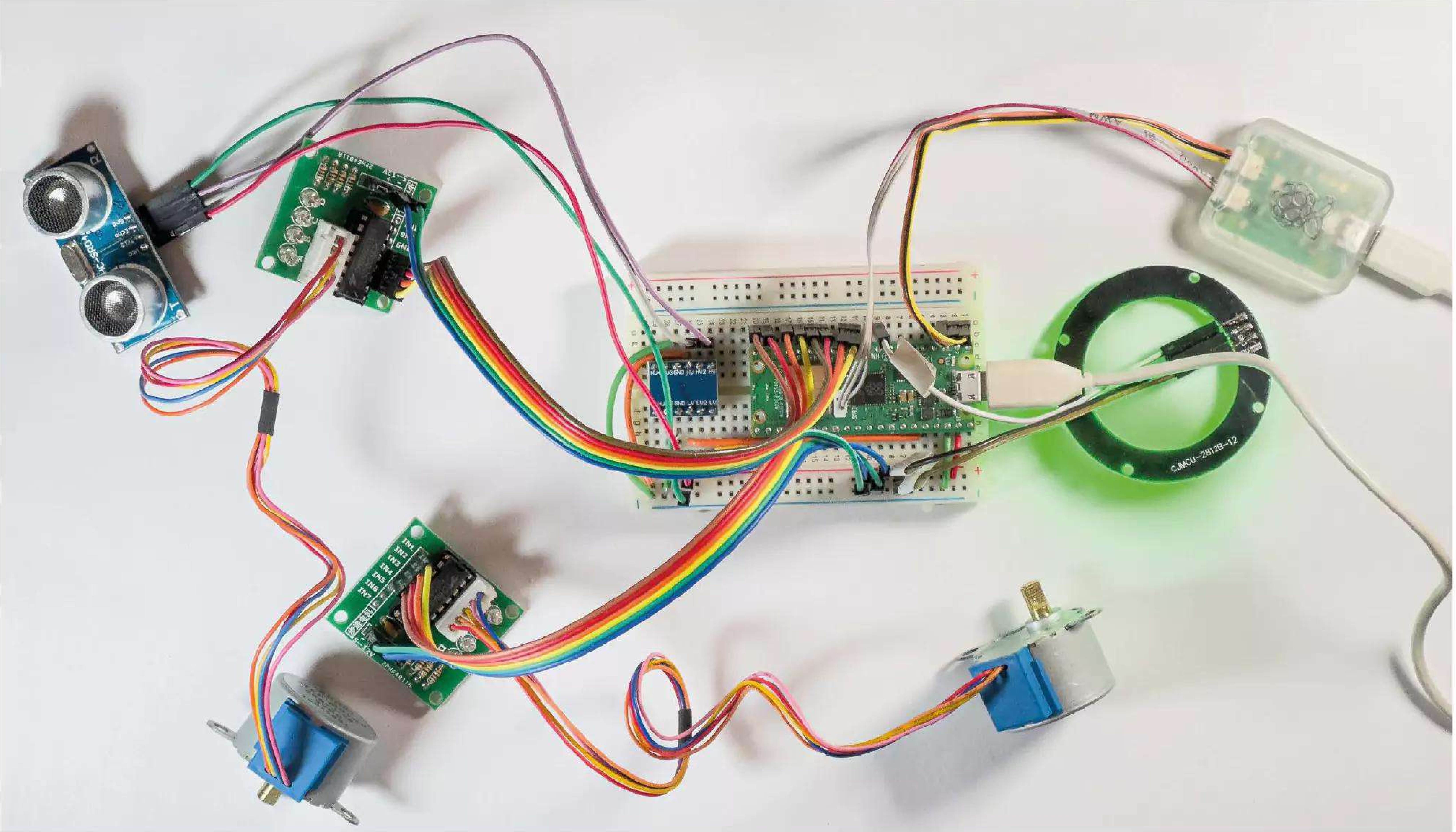


Figure 4
The C++ program that implements Python-ish is being debugged using a Pico Debug Probe. This lets us look inside the program and see what it is doing, even when it is running an interrupt handler

The robot contains two motors, so the above code must be repeated to drive them both. However, we now have a problem. We want the robot to be able to do things while the wheels are turning. This is easy if we are using a DC servo motor that turns when power is applied, but stepper motors require continuously changing signals from the software. If the processor is updating the motor position, it can't do anything else. What we need is a way of generating the above signals without the processor having to wait around. It turns out that there is a way, and it involves using interrupts again.

HARDWARE TIMER TO THE RESCUE

A hardware timer can generate interrupts at a specified interval. Code in a timer interrupt handler can move the motors 'in the background' while the foreground process animates the pixel and runs the program in the robot. The motor driver code calculates the next time that each motor should move to its next step. The robot can move in curves (or the wheels might be of different sizes), so the wheels may be rotating at different speeds. The driver decides which motor needs to be driven first and sets a timer to generate an interrupt at that time. Then, after the interrupt for that step has been performed, it does the same calculation for the next one. The code to make this work reliably was a bit fiddly to write, but the result is motors that move with perfect accuracy without the foreground program doing anything.

```
currentMicros = micros();

if (rightMotorWaveformDelta != 0)
{
  // right is moving - see if it is time to do a step
  rightTimeSinceLastStep =
    ulongDiff(currentMicros, rightTimeOfLastStep);
```

The code above shows how it works. It gets the current time and then decides if the right-hand motor needs to move. There is similar code for the left-hand motor. The variable **rightMotorWaveformDelta** is set to 1 if the motor is moving forwards or -1 if it is moving backwards. So, if **rightMotorWaveformDelta** is non-zero, it might be time to move the right motor. The code reads the current time in microseconds and stores it in the **currentMicros** variable. It stores the difference between the current time and the time when the motor was last moved in the variable **rightTimeSinceLastStep**.

The **ulongDiff** function is used to get this value. It compares two integers and allows for them 'wrapping round'. Values wrapping round is a problem in computer programs. Data values are allocated a certain amount of memory space and if they are given a value which won't fit in this space, they will wrap around. The microsecond counter in the Arduino is held in a 32-bit variable which will wrap around back to zero when it reaches 4,294,967,295. This will happen every 71 minutes or so. The **ulongDiff** function calculates the time value allowing for this wrap round. →

QUICK TIP

We can make the stepper motor run backwards by going through the array in reverse order.

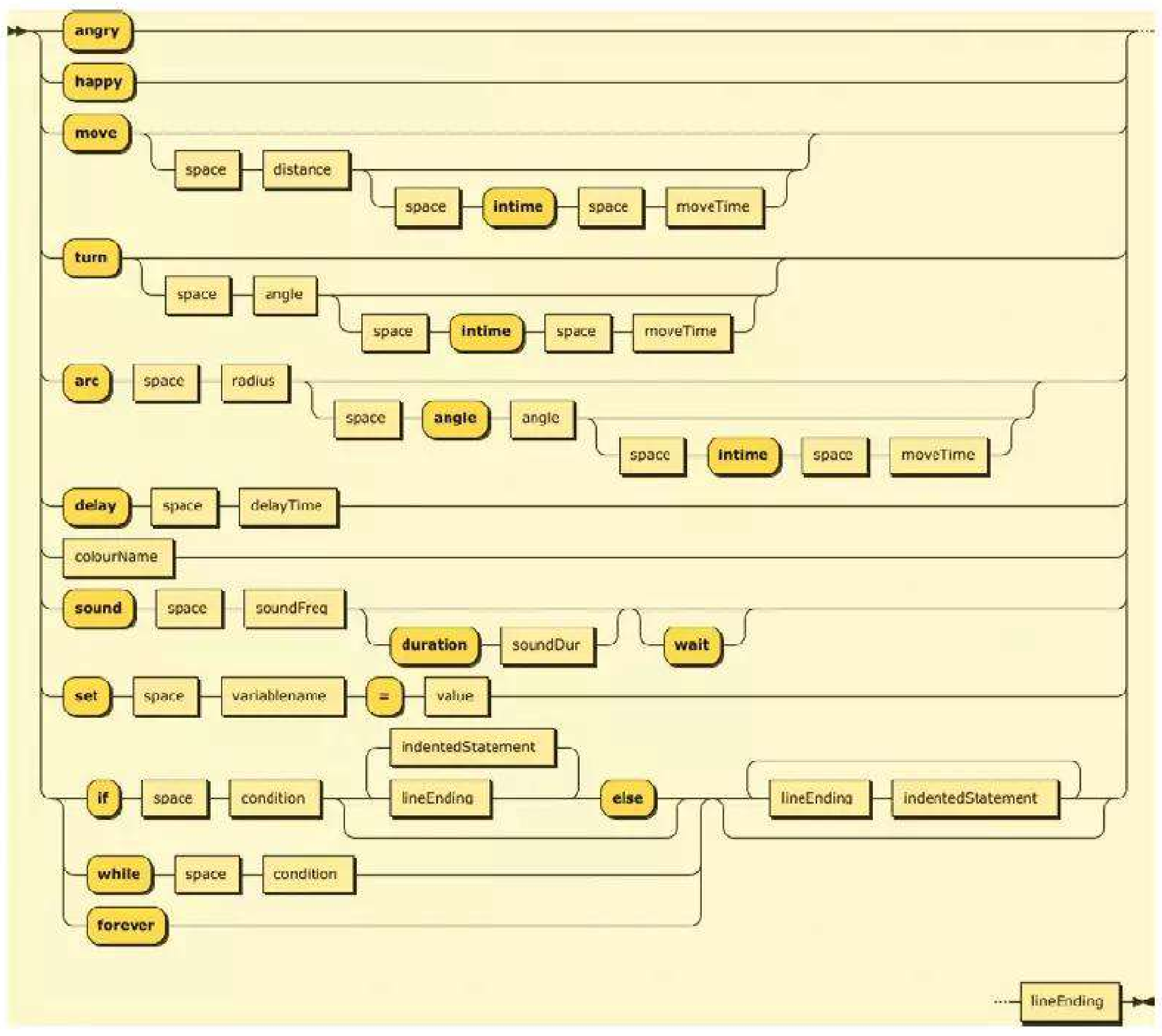


Figure 5 The diagram is created from a grammar which describes the language structure. All the details are in the language description at hullpixelbot.com

```
  if (rightTimeSinceLastStep >=
rightIntervalBetweenSteps)
  {
    rightStep(); // move the motor
    rightTimeOfLastStep = currentMicros -
        (rightTimeSinceLastStep -
rightIntervalBetweenSteps);
    rightTimeOfNextStep = currentMicros +
rightIntervalBetweenSteps;
  }
}
```

Once we have the **`rightTimeSinceLastStep`** value, we can use this to decide if it is time to move the motor. If it is time to move, the **`rightStep`** function is called to move the motor and the time of the next step is set. The same process is performed for the left motor, and then the time of the next interrupt is set depending on whether left or right need to move next. The statement below would run if the right motor must move next:

```
Timer1.setInterval(timeToRight, motorUpdate);
```

The **`motorUpdate`** function checks to see which motor is to be moved and sends motor pulses as required.

PIXEL POWER

The pixel is the simplest part of the robot. The software contains a set of counters which are used to manage the brightness of the colours that are displayed. At regular intervals the colour values are updated. The speed at which they change is controlled by a variable called **`flickerUpdateSpeed`**. The smaller the value, the faster the colours change. The function below sets a new value of the flicker speed, making sure that it can't exceed the limits.

```
int flickerUpdateSpeed = 8;
```

```
void setFlickerUpdateSpeed(byte speed)
{
    // clamp the value
    if (speed > 20)
        speed = 20;
    if (speed < 1)
        speed = 1;

    // set the new flicker update speed
    flickerUpdateSpeed = 21 - speed;

}
```

BUILDING THE ROBOT

Figure 4, overleaf, shows the control circuit under construction. It was tested on the desktop before being fitted onto the robot chassis.

PYTHON-ISH

You use a programming language to tell a computer what to do. Popular ones include C++, Python, and JavaScript. The robot runs a custom language which was developed to fit into the tiny 24K program, 2K memory, and 1K of permanent storage available on the Arduino Uno. The language looks a bit like Python. It contains special commands that can interact with the robot hardware.

```
# Don't push me!
forever
  d = @distance
  if d < 100:
    angry
    red
  else
    happy
    green
```

The program above can be loaded into the robot and executed. If you are more than 100 mm away from the robot, it shows a happy, green persona. However, get too close and it turns red and angry. The **`forever`** statement repeats a sequence of statements indefinitely. The **`@distance`** special variable returns the latest reading of the distance sensor and the words **`angry`**, **`happy`**, **`green`**, and **`red`** do exactly what you would expect. Like Python, the language uses indenting to indicate nesting. All the statements indented under the **`if`** condition are controlled by it.

The Python-ish compiler converts the text above into a sequence of two-character commands. This

QUICK TIP

There are commands to set the sizes and spacing of the two driving wheels, so you can use whatever sizes you like. You can even make a robot with differently-sized wheels which will still move in a straight line. The size values are retained in the robot. The command to set the sizes is:

`*MWll,rr,www`

'll' is the diameter of the left-hand wheel in mm, 'rr' is the diameter of the right-hand wheel, and 'www' is the distance between the wheels.

is achieved by extracting individual words from the program source and then looking the words up in a long string:

```
// command numbers:            0     1     2     3...
const byte commandNames[] = "angry#happy#move#turn...
```

The **commandNames** string contains all the 'keywords' for the Python-ish language. Each keyword is separated from the next by a # character. There are 111 keywords, including lots of colour names. The program searches through this string for a command and notes the number of the command it finds. These numbers are defined in the program.

```
#define COMMAND_ANGRY 0
#define COMMAND_HAPPY 1
#define COMMAND_MOVE 2
#define COMMAND_TURN 3
```

Now that the compiler knows which command it is dealing with, it can perform a C++ switch to select the code to deal with that command:

```
int processCommand(byte commandNo)
{
    switch (commandNo)
    {
    case COMMAND_ANGRY:// angry
        return compileAngry();

    case COMMAND_HAPPY:// happy
        return compileHappy();
    ...
```

The compile function for a given language element processes the language element and generates a piece of program code which will perform the command that was recognised.

```
const byte angryCommand[] PROGMEM = "PF20";

int compileAngry()
{
#ifdef SCRIPT_DEBUG
    Serial.println(F("Compiling angry: "));
#endif // SCRIPT_DEBUG

    sendCommand(angryCommand);
    previousStatementStartedBlock = false;
    return ERROR_OK;
}
```

The **compileAngry** function uses the **sendCommand** function to send **"PF20"** to the sequence of compiled commands. The letter **P** means 'pixels' and the letter **F** means 'flickerspeed'. The value that follows sets the flicker speed to 20 so that the lights on the robot will flicker angrily. The **compileHappy** function generates the sequence **"PF1"** which makes the lights update in a much more chilled manner. Some of the compiled commands generate more complicated command sequences, but the fundamental principle applies to all the code.

Each 'compiled' statement starts with two characters. The first specifies the 'family' of the command (**P** for pixel, **V** for variable, **C** for control →

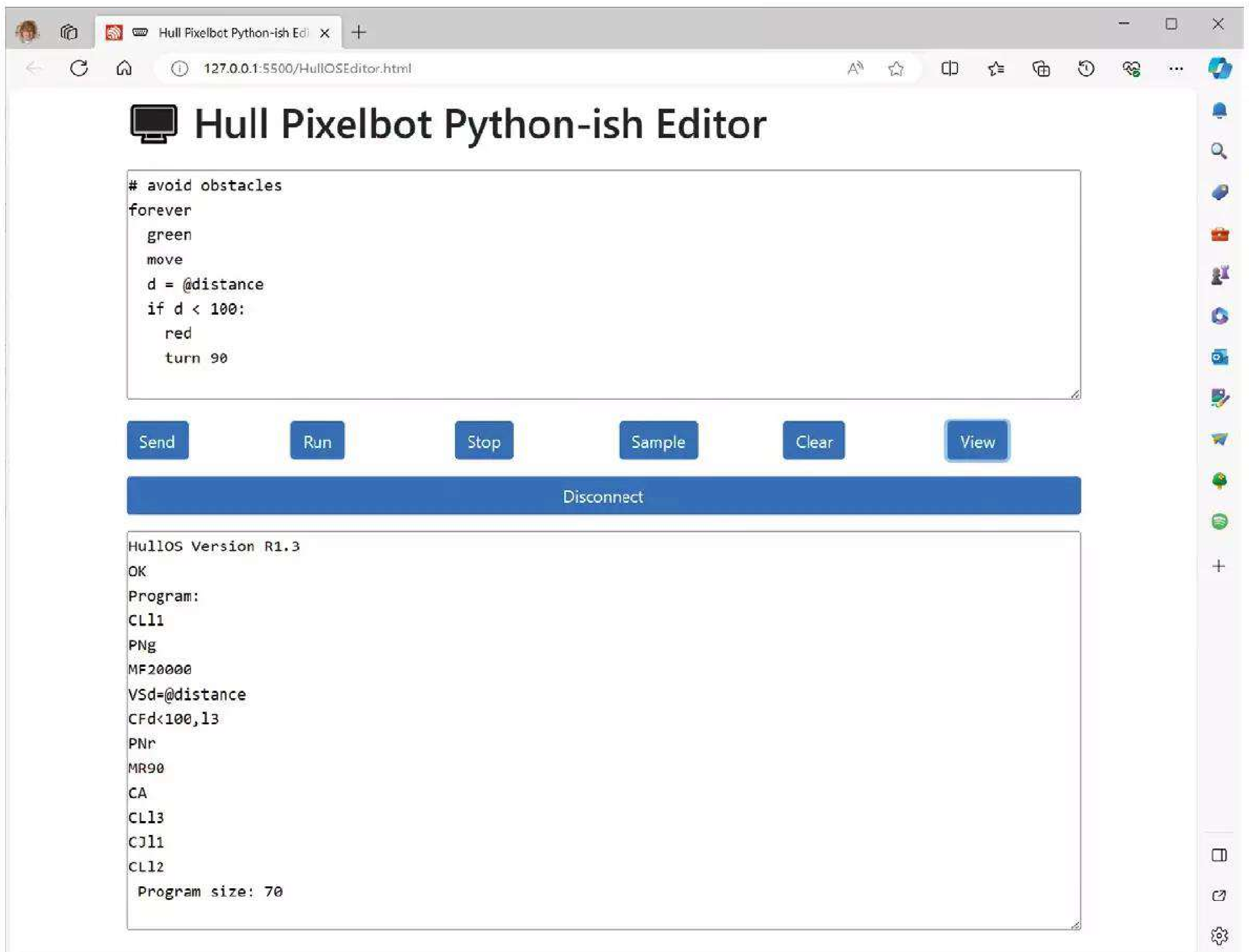
A BIT SLIPPERY

Python-ish was created to fit inside the tiny processor in the Arduino Uno. You may be wondering if we need to retain it when using the much more powerful and spacious Pico or ESP32 devices. Why not switch to CircuitPython or MicroPython and take advantage of all the features in those languages to tell our robots what to do? If you wish, you can do this, but the Python-ish language does have advantages. The robot can obey lines of code even when it is running another program. This is particularly useful when the robot is controlled remotely. And the programs themselves can execute multiple concurrent processes and bind behaviours directly to language elements. And because the language does less, it is much easier to learn.

QUICK TIP

Once the circuit had been tested, the author spread hot glue around all the pins going into the breadboard. This made the robot much more resilient.

Below There is a link to the online version of the editor at hullpixelbot.com


Hull Pixelbot Python-ish Edi
127.0.0.1:5500/HullOSEditor.html
Hull Pixelbot Python-ish Editor
Send
Run
Stop
Sample
Clear
View
Disconnect

```
# avoid obstacles
forever
  green
  move
  d = @distance
  if d < 100:
    red
    turn 90
```

```
HullOS Version R1.3
OK
Program:
CLl1
PNg
MF20000
VSd=@distance
CFd<100,l3
PNr
MR90
CA
CLl3
CJl1
CLl2
 Program size: 70
```

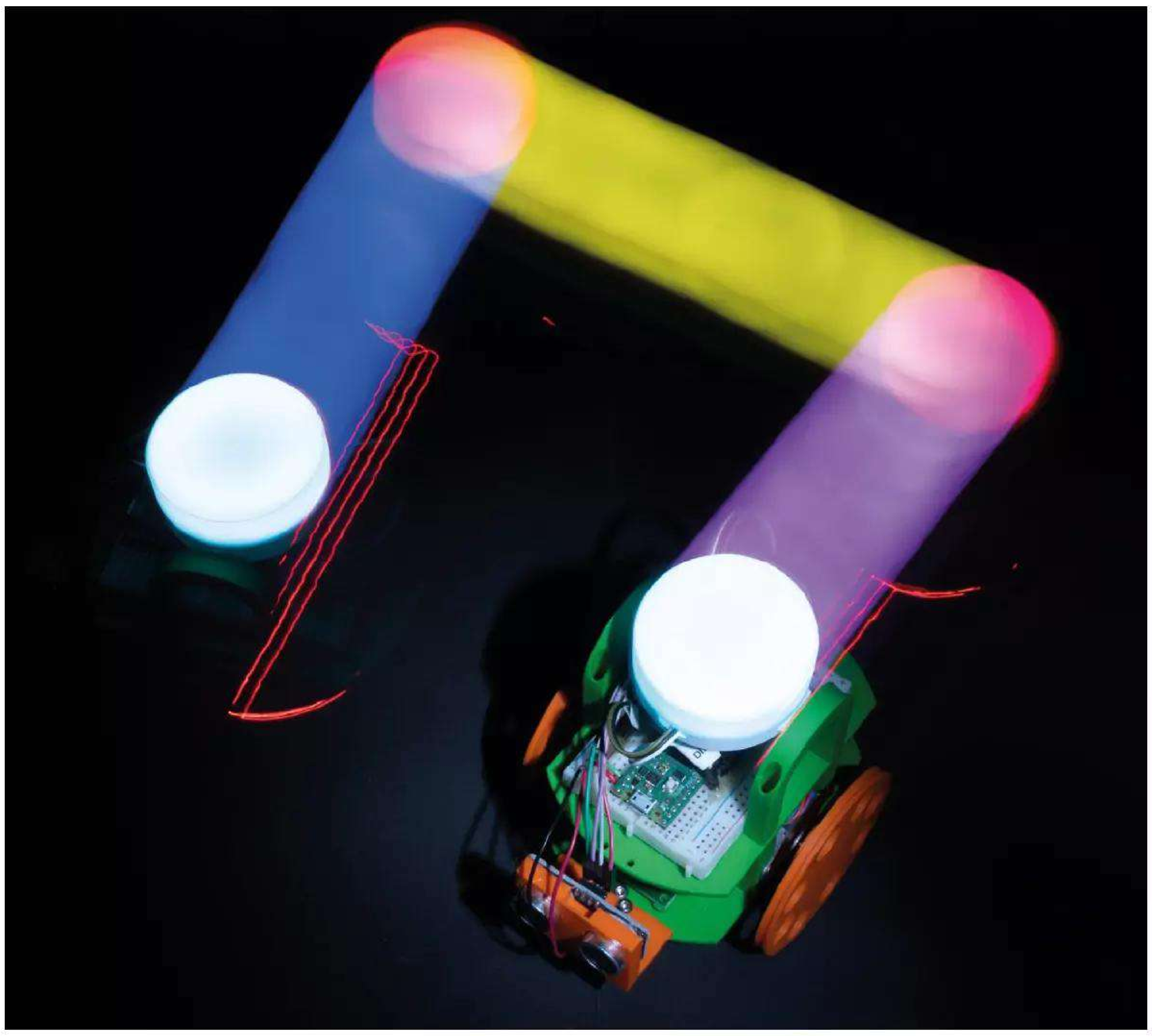


Figure 6 The red trails are from the stepper motor LEDs

etc.) and the second character gives the command itself (**F** for flickerspeed in **PF** or **S** for variable set in **VS**). The author could have used numbers for each compiled command, but he wanted to be able to understand the compiled code. The code below is the compiled output for the Python-ish code above. The command **CL** creates a label, the command **CJ** performs a jump to a label, and **CF** performs a jump if the following condition is false.

```
CLl1
VSd=@distance
CFd<100,l3
PF20
PNr
CJl4
CLl3
PF1
PNg
CLl4
CJl1
CLl2
```

If you look carefully at the statements (and draw some lines connecting jumps to their destinations), you can start to see how the high-level language of Python-ish has been converted into a sequence of low-level steps. When the program runs, it uses more switch statements to pick the code to perform each command. Note that at no point does the program actually create any Arduino Uno machine code – the run time system is an 'interpreter' which just takes a set of instructions and does what they say.

RAILROADS AND LANGUAGE DESIGN

The final part of the language design is the description of the language syntax. The author could have written a bunch of text describing how you can write the programs, but instead he has created some 'railroad diagrams'. **Figure 5** shows the diagram describing what you can use in a statement. You can use this to work out what you can and can't do in the language. For example, it shows that a **move** statement can optionally be followed by a space (one or more spaces) and a **distance** value. However, a **delay** statement must be followed by a space and a **delaytime** value. Making these diagrams is quite fun (at least for the author). You can find the Railroad Diagram generator at **bottlecaps.de/rr/ui**.

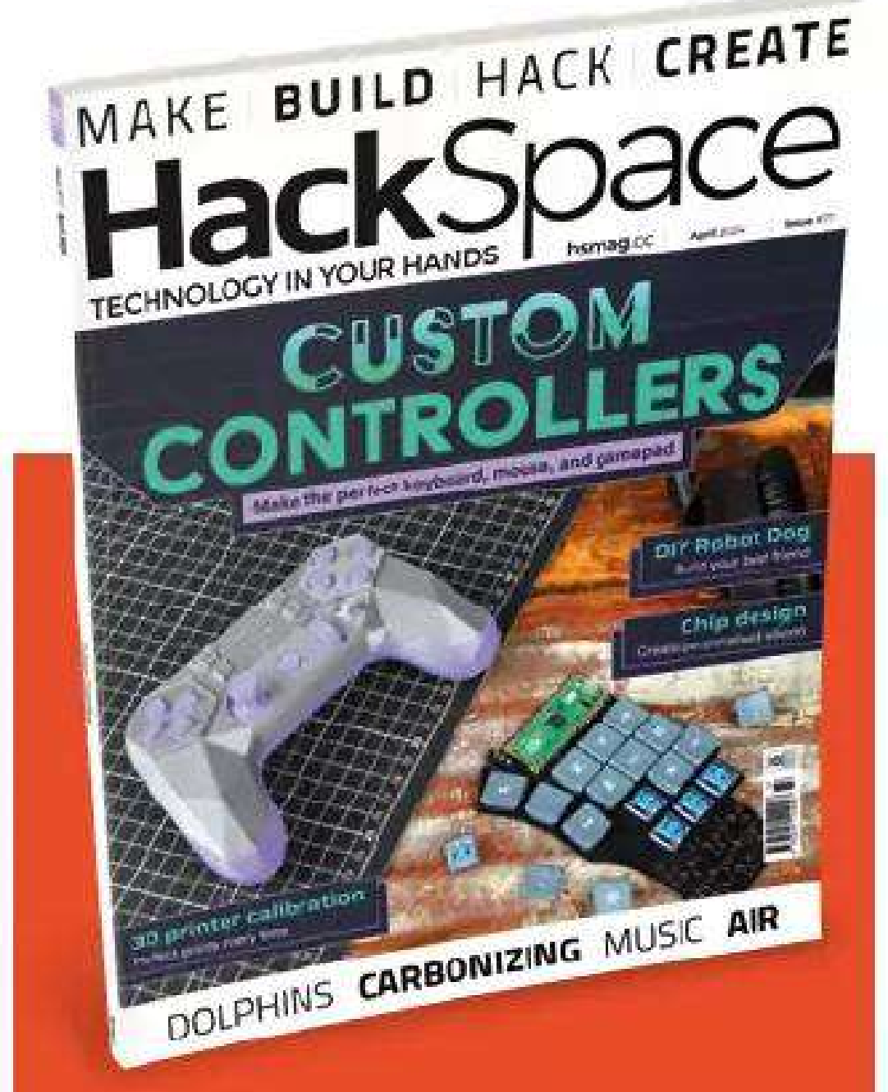
> "The robot is fun to play with, but it is inconvenient **to have to keep connecting it to a computer**"

EDITING PYTHON-ISH PROGRAMS

You can edit programs in your robot by plugging it into a serial port on your computer and then opening the program edit page in your browser. You can send your programs into the robot and run and stop them. The editor also has some built in sample programs. The Clear button removes the program from the robot and the View button displays the low-level code in the robot. You can find the editor online at **hullpixelbot.com/Python-ish**.

FURTHER DEVELOPMENT

Figure 6 shows a robot on the move. The robot is fun to play with, but it is inconvenient to have to keep connecting it to a computer to program it. In the next article in this series, we'll discover how you can add a network connection to your robot and enter programs remotely. This makes all kinds of fun things possible, including 'robot rugby' where players program their Pixelbots to take on their opponents. □



HackSpace

This tutorial is from HackSpace magazine. Each issue includes a huge variety of maker projects inside and outside of the sphere of Raspberry Pi, and also has amazing tutorials. Find out more at **hsmag.cc**

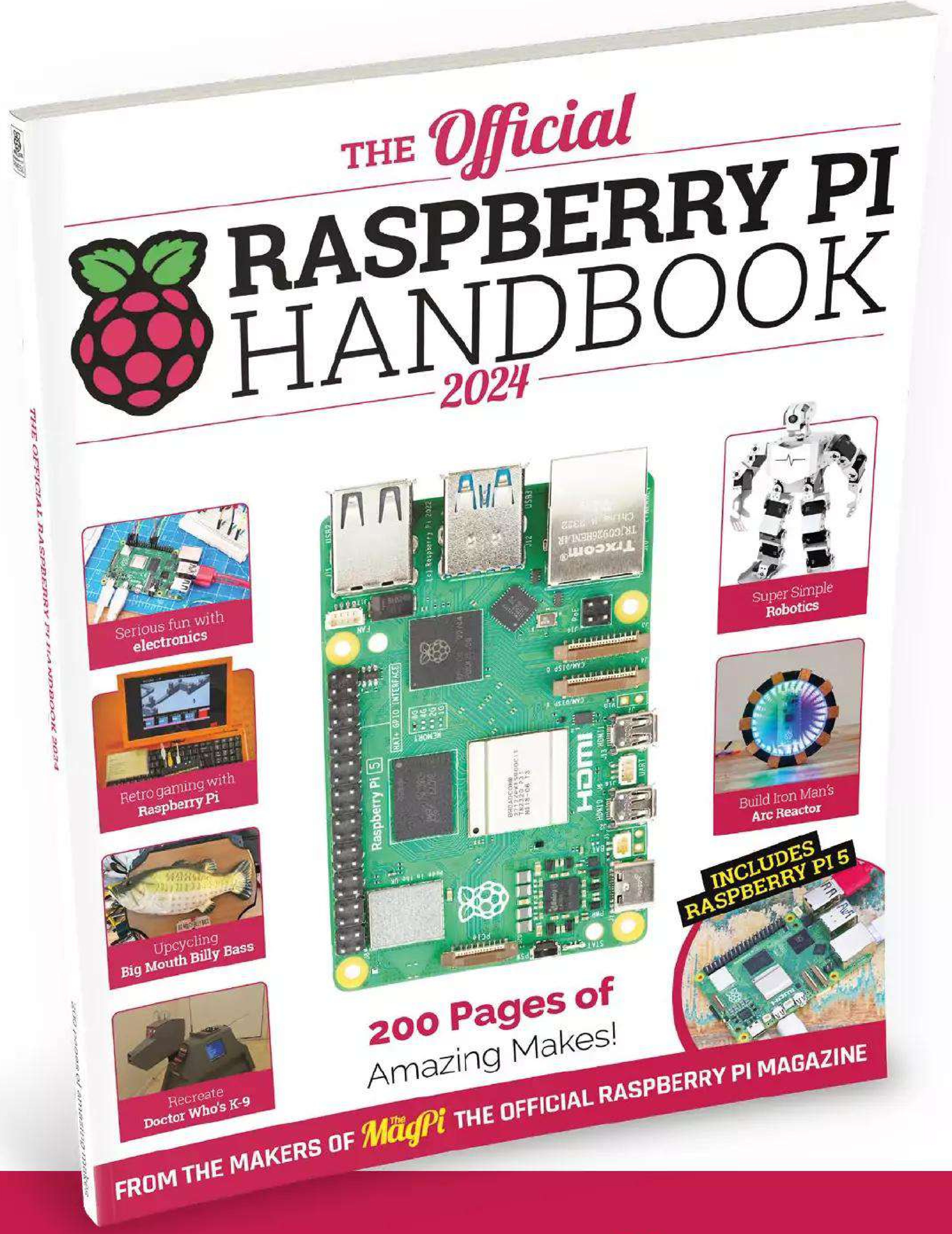
THE *Official*

RASPBERRY PI HANDBOOK

2024

200 PAGES OF RASPBERRY PI

- QuickStart guide to setting up your Raspberry Pi computer
- Updated with Raspberry Pi Pico and all the latest kit
- The very best projects built by your Raspberry Pi community
- Discover incredible kit and tutorials for your projects



THE Official
RASPBERRY PI
HANDBOOK
2024
Serious fun with electronics
Retro gaming with Raspberry Pi
Upcycling Big Mouth Billy Bass
Recreate Doctor Who's K-9
Super Simple Robotics
Build Iron Man's Arc Reactor
INCLUDES RASPBERRY PI 5
200 Pages of Amazing Makes!
FROM THE MAKERS OF MagPi THE OFFICIAL RASPBERRY PI MAGAZINE

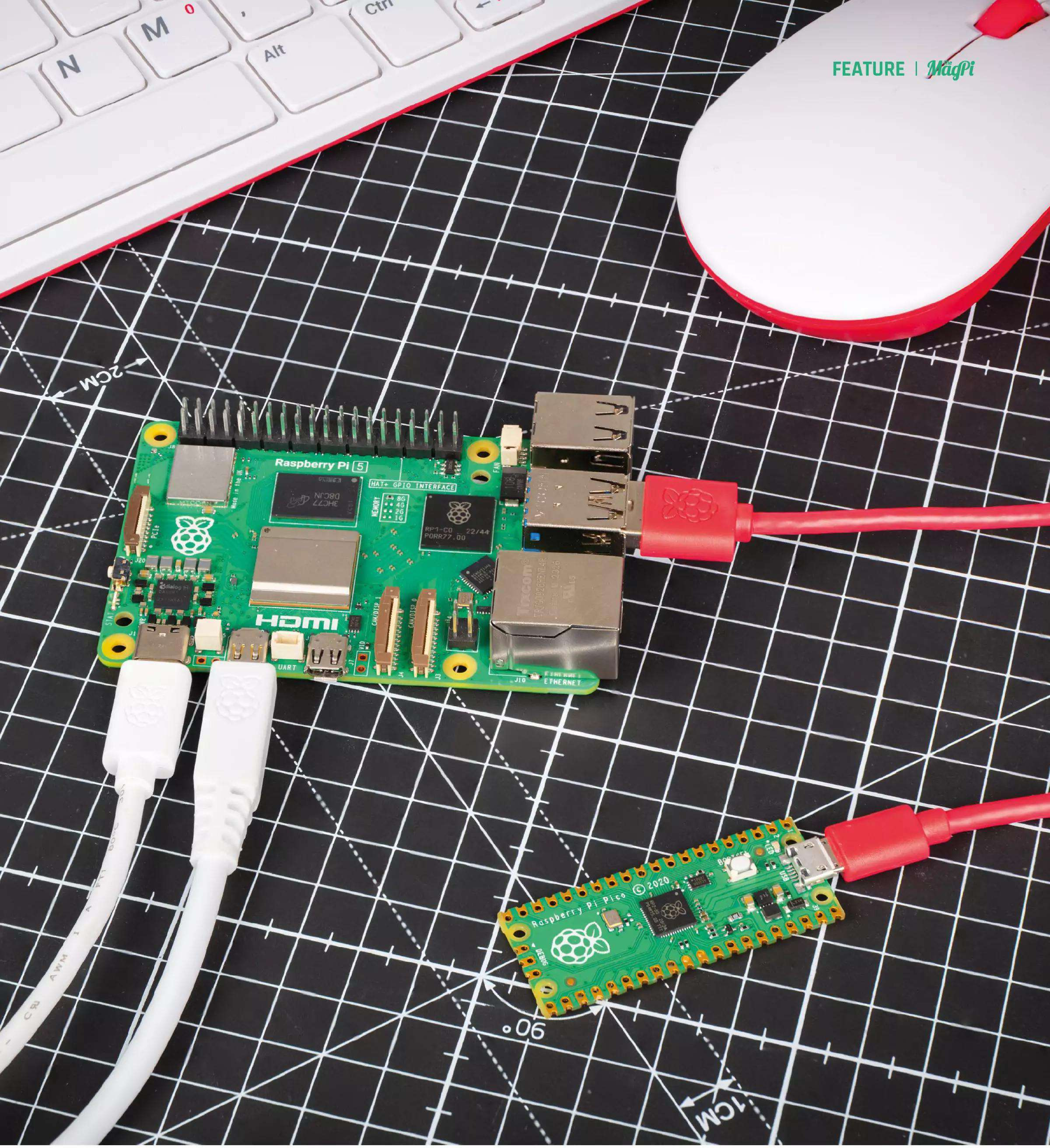
Buy online: magpi.cc/store

Raspberry Pi 5 & Raspberry Pi Pico

by Phil King

By connecting Raspberry Pi 5 to Raspberry Pi Pico, you get the best of both worlds: a single-board computer linked to a low-cost, compact microcontroller. Pico can be programmed in the MicroPython language using the Thonny IDE on Raspberry Pi 5. It is then able to run a program automatically when powered up on its own, so can be used in all sorts of portable projects.

Here we'll show you how to set up Pico with Raspberry Pi 5, install the MicroPython firmware on it, and write a program to flash Pico's on-board LED. We'll also take a look at the optional Debug Probe which can be used to see exactly what Pico's processor is doing.


Raspberry Pi 5
HDMI
Raspberry Pi Pico

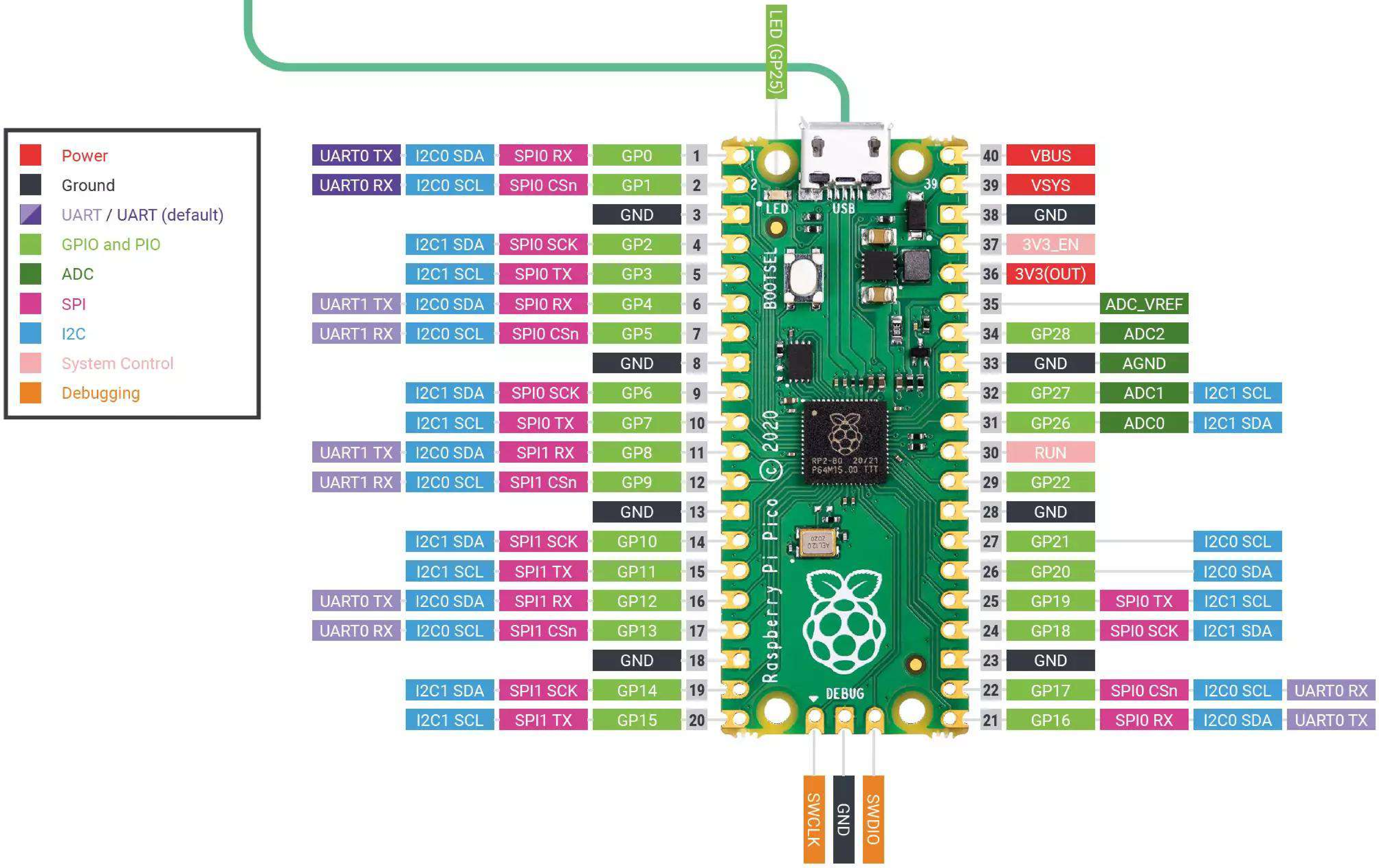
Introducting **Raspberry Pi Pico**

Raspberry Pi Pico is an inexpensive compact microcontroller board that can run programs with very low power drain, making it ideal for portable projects.

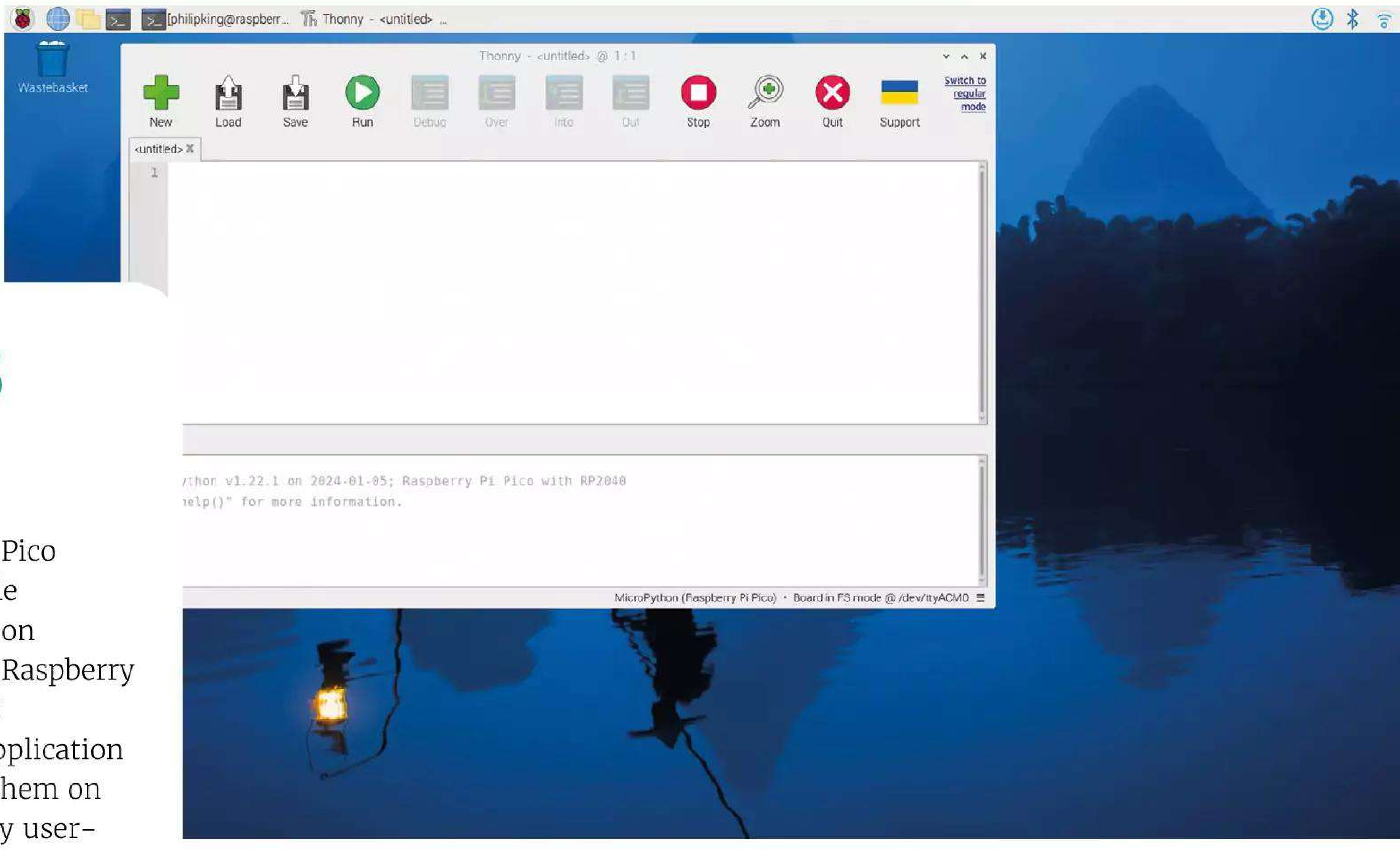
While not as powerful as a Raspberry Pi single-board computer, it is based around an RP2040 microcontroller chip with two Arm Cortex M0+ cores running at up to 133MHz, and unique PIO (Programmable Input/Output) state machines that can run code independently of the main CPU. Pico also features 264kB of SRAM, 2MB of on-board flash memory, and W models have Wi-Fi.

Just like any other Raspberry Pi board, Pico has a set of 40 GPIO pins, although here they are arranged in rows on either side of the board (see 'Pico Pinout' box for details). As well as enabling connection of a large range of third-party add-on boards, the GPIO pins can be used to connect your own circuits comprising standard electronic components such as LEDs, buttons, sensors, and motors – so you could use Pico in a robot.

Pico can be programmed from a USB-connected computer, such as a Raspberry Pi, using the easy-to-learn MicroPython language or, for more advanced users, C/C++. Pico can then run a program automatically when powered up, so you can use it when disconnected from the computer.



Power
Ground
UART / UART (default)
GPIO and PIO
ADC
SPI
I2C
System Control
Debugging
LED (GP25)
UART0 TX | I2C0 SDA | SPI0 RX | GP0 | 1
UART0 RX | I2C0 SCL | SPI0 CSn | GP1 | 2
GND | 3
I2C1 SDA | SPI0 SCK | GP2 | 4
I2C1 SCL | SPI0 TX | GP3 | 5
UART1 TX | I2C0 SDA | SPI0 RX | GP4 | 6
UART1 RX | I2C0 SCL | SPI0 CSn | GP5 | 7
GND | 8
I2C1 SDA | SPI0 SCK | GP6 | 9
I2C1 SCL | SPI0 TX | GP7 | 10
UART1 TX | I2C0 SDA | SPI1 RX | GP8 | 11
UART1 RX | I2C0 SCL | SPI1 CSn | GP9 | 12
GND | 13
I2C1 SDA | SPI1 SCK | GP10 | 14
I2C1 SCL | SPI1 TX | GP11 | 15
UART0 TX | I2C0 SDA | SPI1 RX | GP12 | 16
UART0 RX | I2C0 SCL | SPI1 CSn | GP13 | 17
GND | 18
I2C1 SDA | SPI1 SCK | GP14 | 19
I2C1 SCL | SPI1 TX | GP15 | 20
40 | VBUS
39 | VSYS
38 | GND
37 | 3V3_EN
36 | 3V3(OUT)
35 | ADC_VREF
34 | GP28 | ADC2
33 | GND | AGND
32 | GP27 | ADC1 | I2C1 SCL
31 | GP26 | ADC0 | I2C1 SDA
30 | RUN
29 | GP22
28 | GND
27 | GP21 | I2C0 SCL
26 | GP20 | I2C0 SDA
25 | GP19 | SPI0 TX | I2C1 SCL
24 | GP18 | SPI0 SCK | I2C1 SDA
23 | GND
22 | GP17 | SPI0 CSn | I2C0 SCL | UART0 RX
21 | GP16 | SPI0 RX | I2C0 SDA | UART0 TX
SWCLK
GND
SWDIO
LED
USB
BOOTSEL
DEBUG
Raspberry Pi Pico © 2020



```
1
```

```
/thon v1.22.1 on 2024-01-05; Raspberry Pi Pico with RP2040
ıelp()" for more information.
```

MicroPython (Raspberry Pi Pico) • Board in FS mode @ /dev/ttyACM0

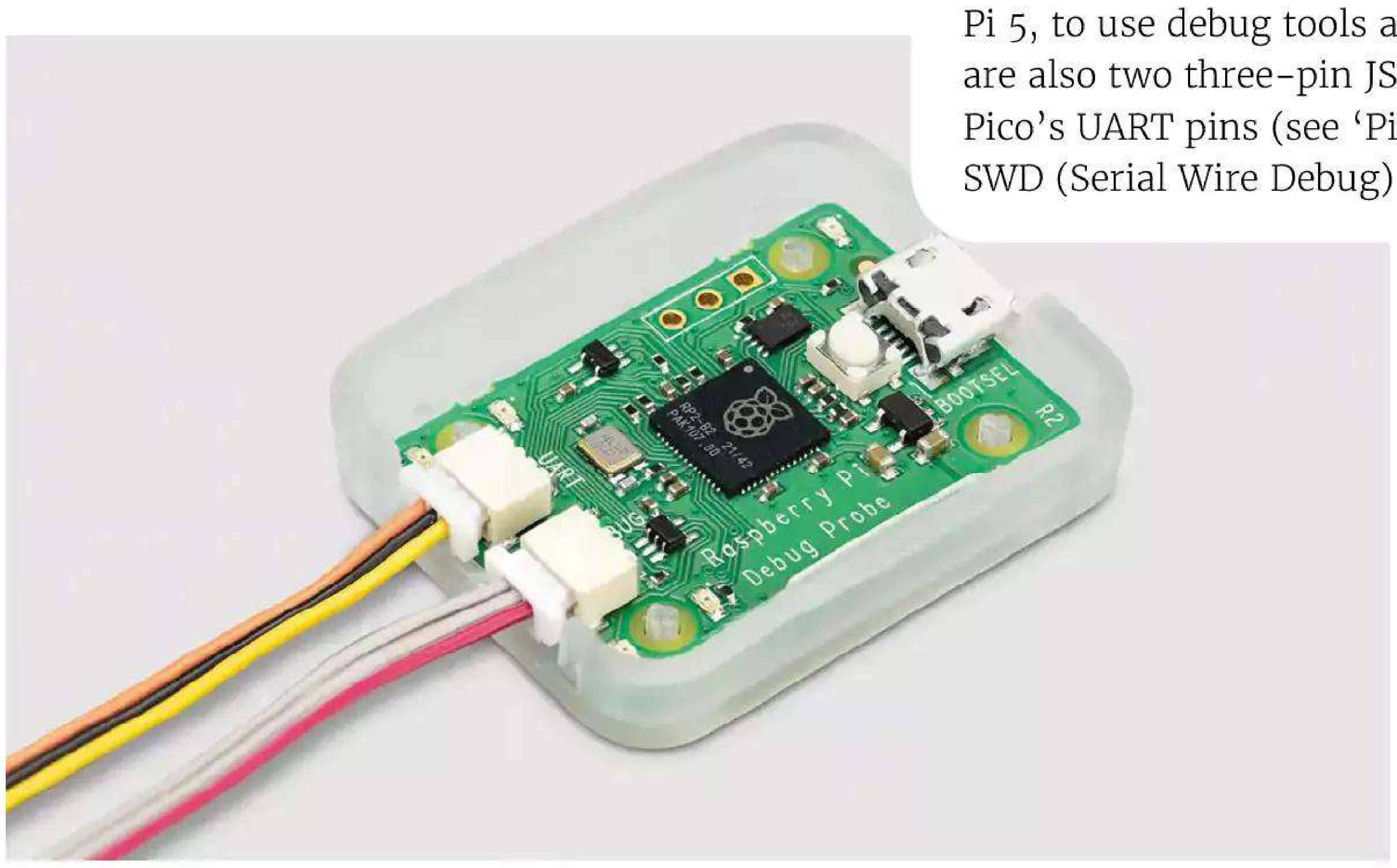
Raspberry Pi 5 and Thonny IDE

The easiest way to program Raspberry Pi Pico is with MicroPython, a Python-compatible programming language developed to run on microcontrollers. With Pico connected to Raspberry Pi 5 via USB, you can use the Thonny IDE (integrated development environment) application to write MicroPython programs and run them on Pico. Packed with features, Thonny is very user-friendly and its 'Shell' pane will show you any output from the running Pico program, or error messages if something goes wrong. It also features a built-in debugger which lets you walk through a program step by step to show what's happening at each stage, helping you to find bugs.

Debug Probe

While not essential for programming Pico, especially when using MicroPython, this optional add-on enables advanced users to see exactly what Pico is doing, via a serial console. It's especially useful when attempting to debug C/C++ code running directly on Pico's RP2040 processor.

Powered by an RP2040 itself, the Debug Probe has three connectors. The micro-USB port is for connecting it to a host computer, such as Raspberry Pi 5, to use debug tools and view the output. There are also two three-pin JST ports, used to connect to Pico's UART pins (see 'Pico Pinout' box) and/or its SWD (Serial Wire Debug) header.



Top Tip

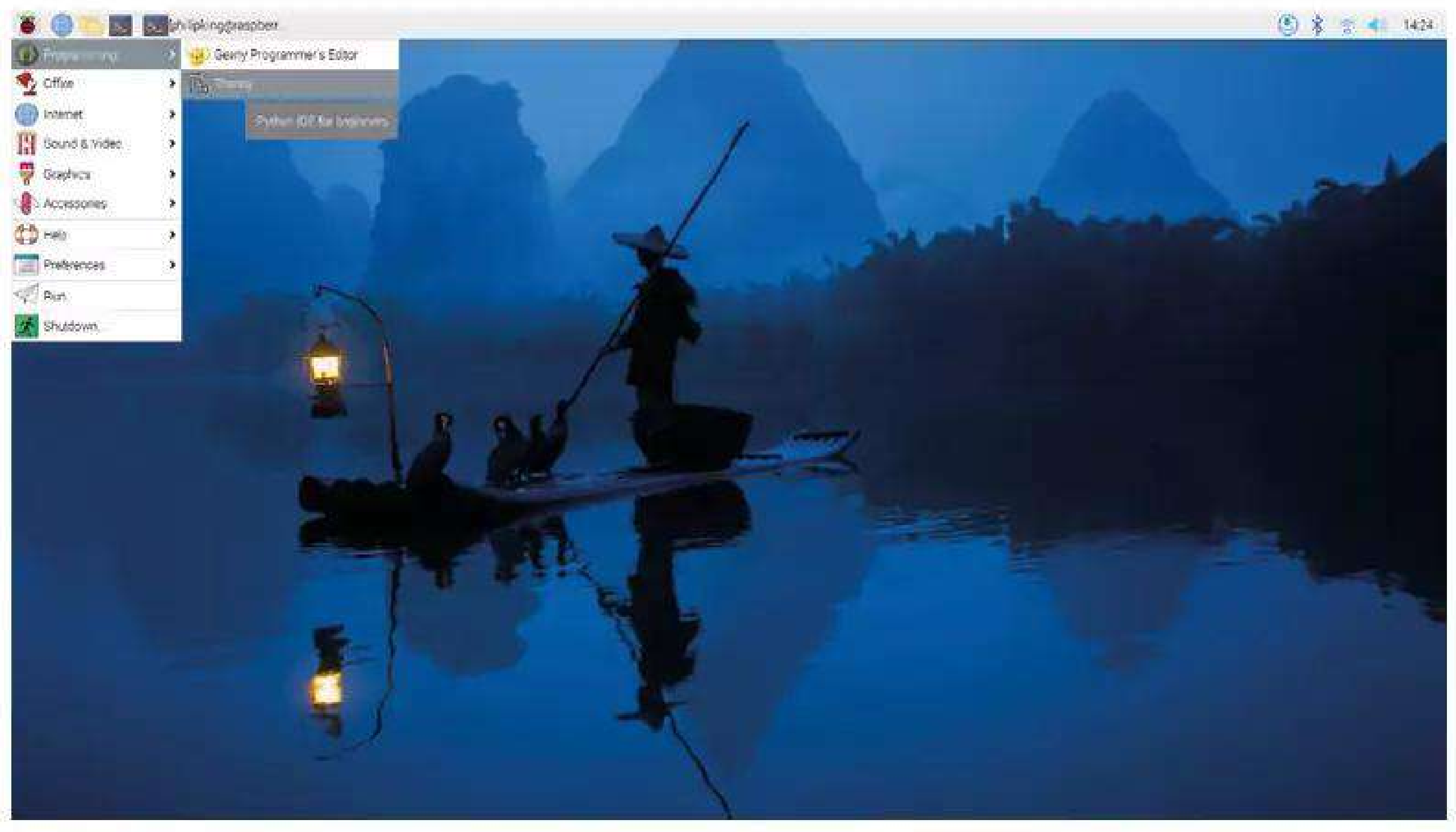
SWD Header

The type and position of the SWD header varies on different Pico models. On Pico and Pico W, it comprises three pins. On Pico H and Pico WH, it's a JST port.

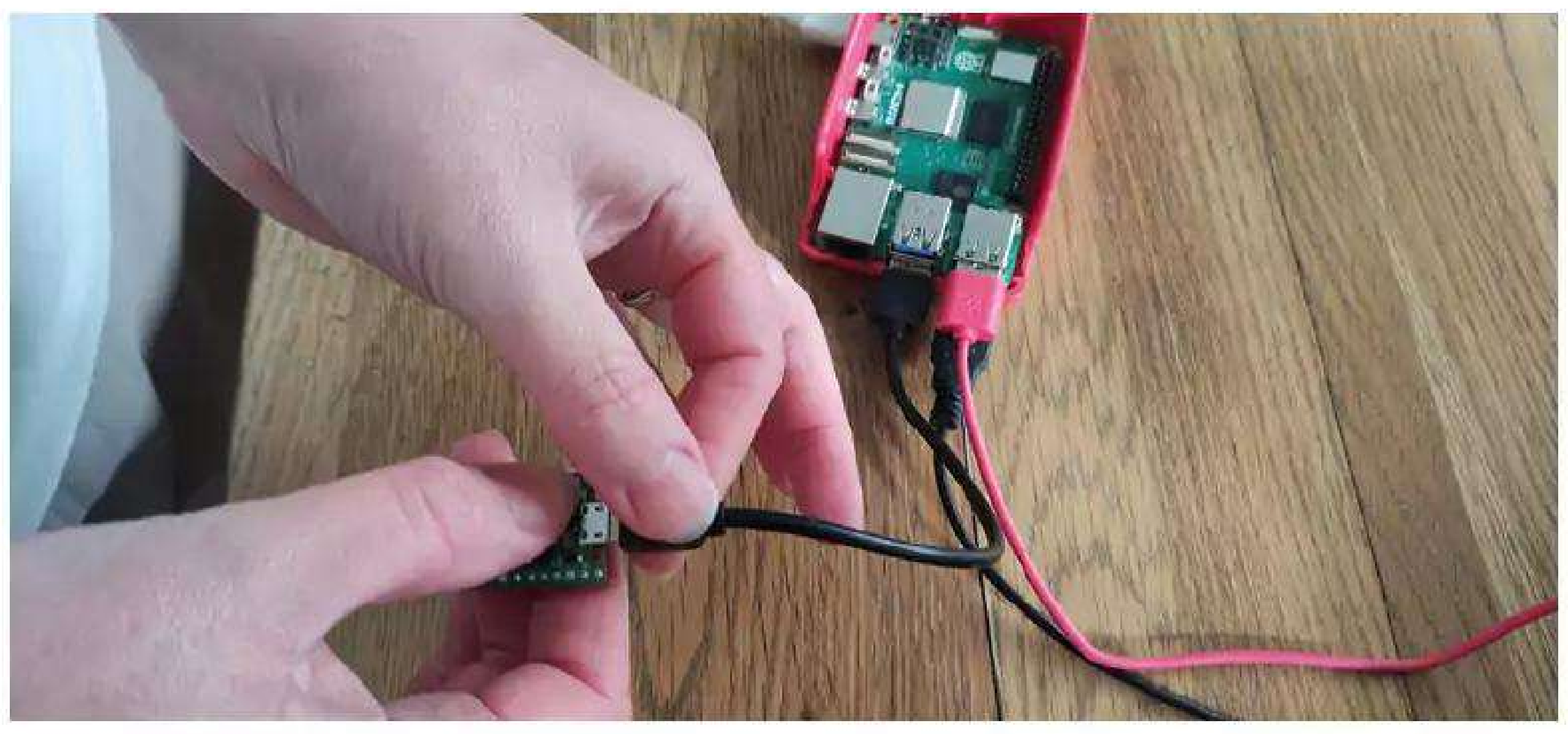
Programming **Pico**

We'll connect Pico to Raspberry Pi 5 and write a MicroPython program to blink its on-board LED.

01 To program Pico using MicroPython, we'll use the Thonny IDE. With Raspberry Pi 5 connected to a monitor, open Thonny – found in (top left) Menu > Programming.



02 Connect the larger end of the cable to any free USB Type A port on Raspberry Pi 5. While holding Pico's BOOTSEL button, connect the other end of the cable to Pico's micro-USB port to mount it as a drive.



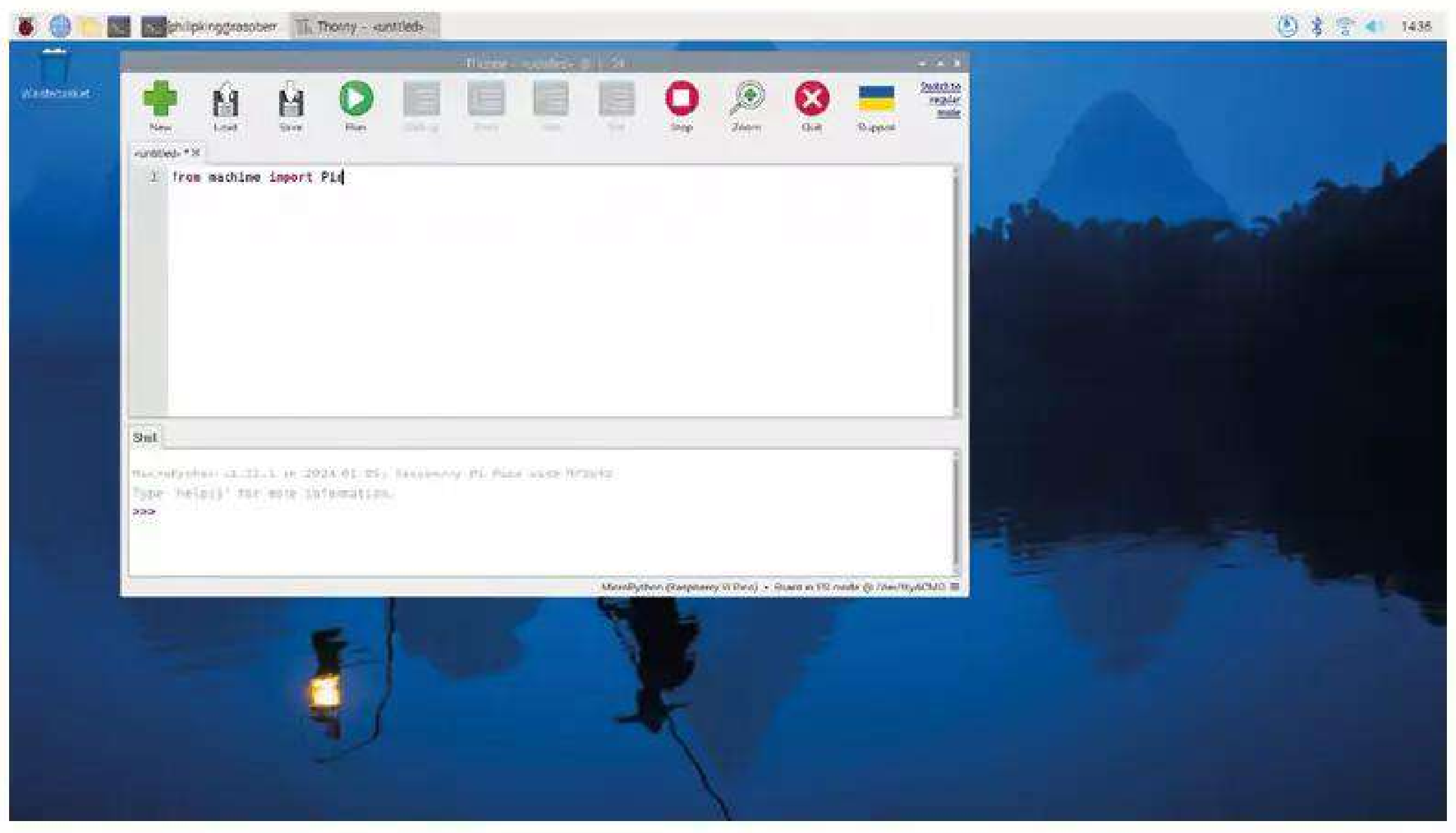
03 We need to install the MicroPython firmware on Pico. In Thonny, click 'Local Python 3' at the bottom right of the window. Select 'Install MicroPython' from the pop-up menu.

04 In the next menu, click the 'variant' drop-down and select your Pico model type to get the correct version. Finally, click the 'Install' button to flash the MicroPython firmware to Pico.

05 A message in the Shell pane will confirm the version of MicroPython installed on the connected Pico. You are now ready to start programming it.

06 To use Pico's GPIO pins, we need to import the Pin class of the `machine` MicroPython module. Add this code to Thonny's main pane:

```
from machine import Pin
```



```
from machine import Pi
```

```
MicroPython v1.22.1 on 2024-01-05; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
```

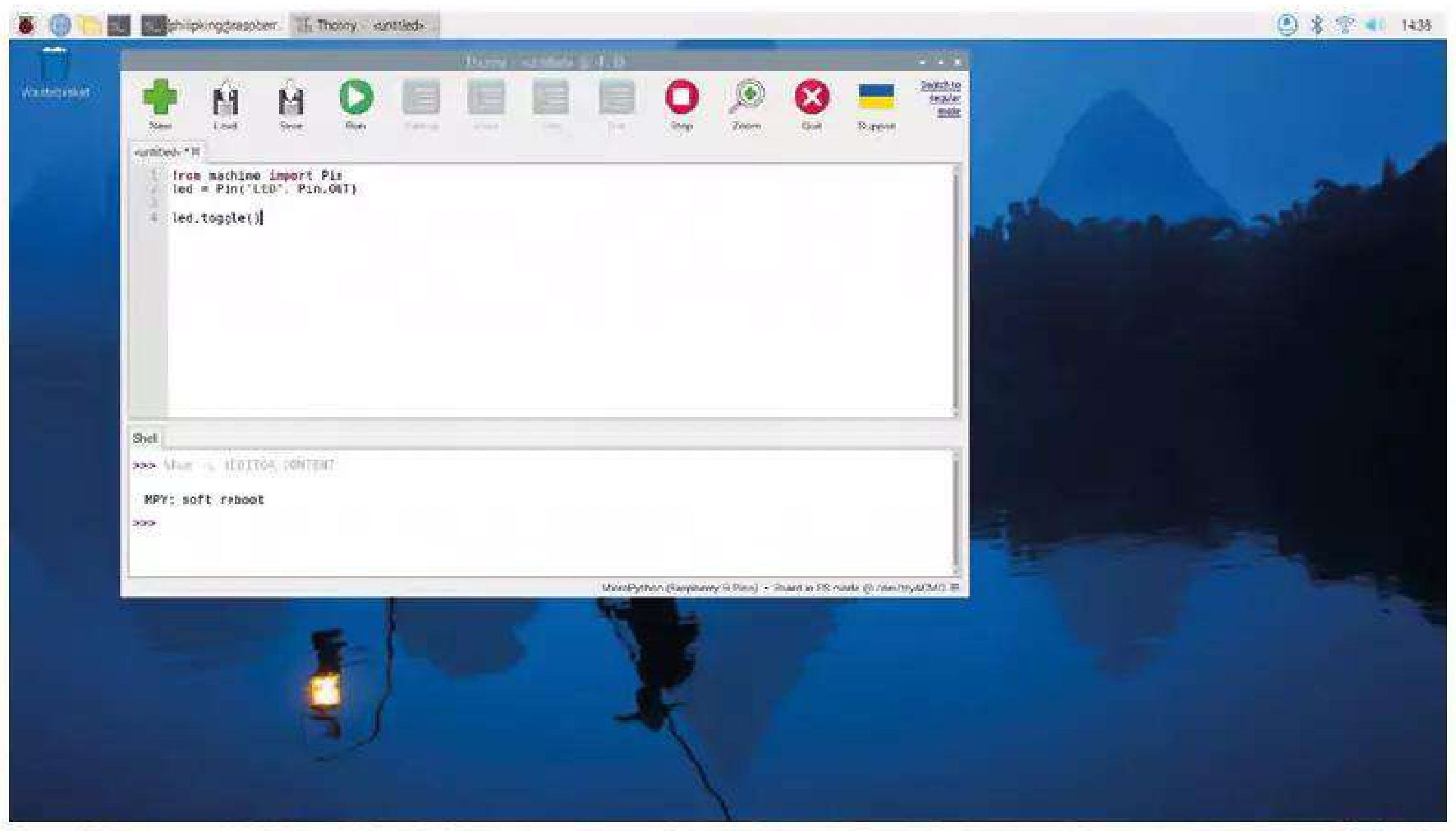
07 Next, we assign a variable (we've named it `led`) to the LED's GPIO pin on Pico and set it as an output:

```
led = Pin("LED", Pin.OUT)
```

08 To toggle the LED on/off, we add the line:

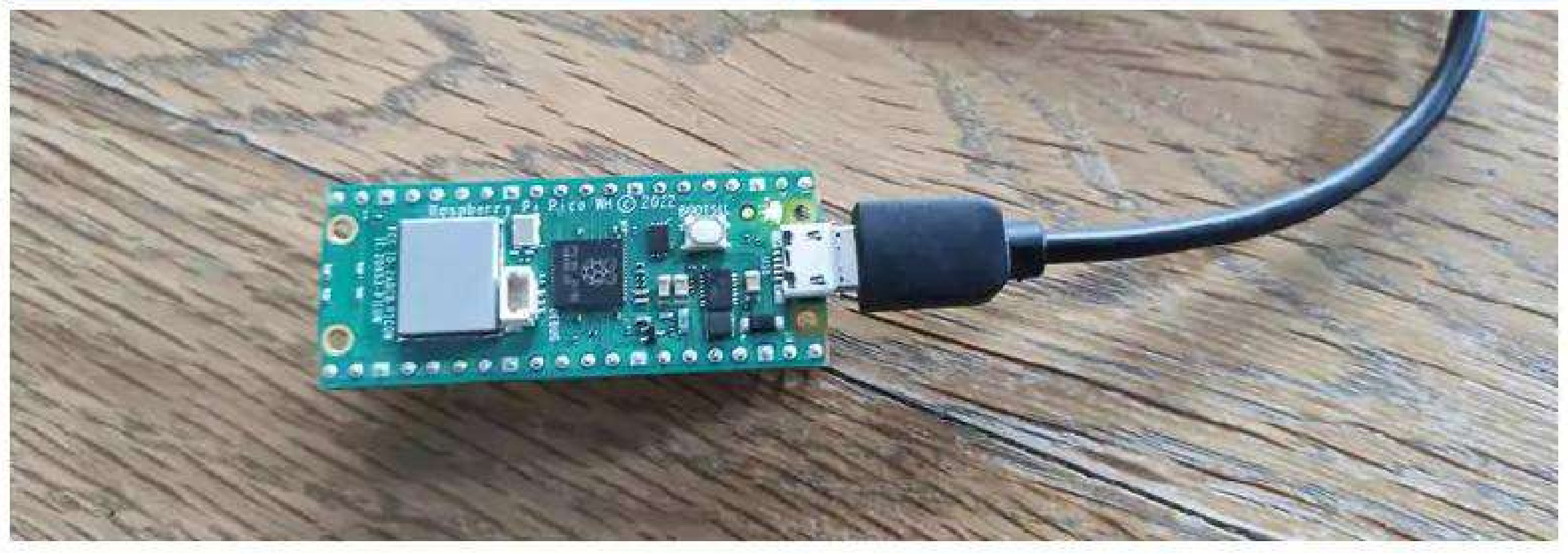
```
led.toggle()
```

Click Thonny's Run button to run the program and turn Pico's LED on. Run it again to turn it off.



```
from machine import Pin
led = Pin('LED', Pin.OUT)

led.toggle()
```

```
>>> %Run -c $EDITOR_CONTENT

 MPY: soft reboot
>>>
```



09 To save your program, click Thonny's Save button and name the file with a .py suffix – we called ours **blink.py** – and opt to save it on Raspberry Pi Pico.

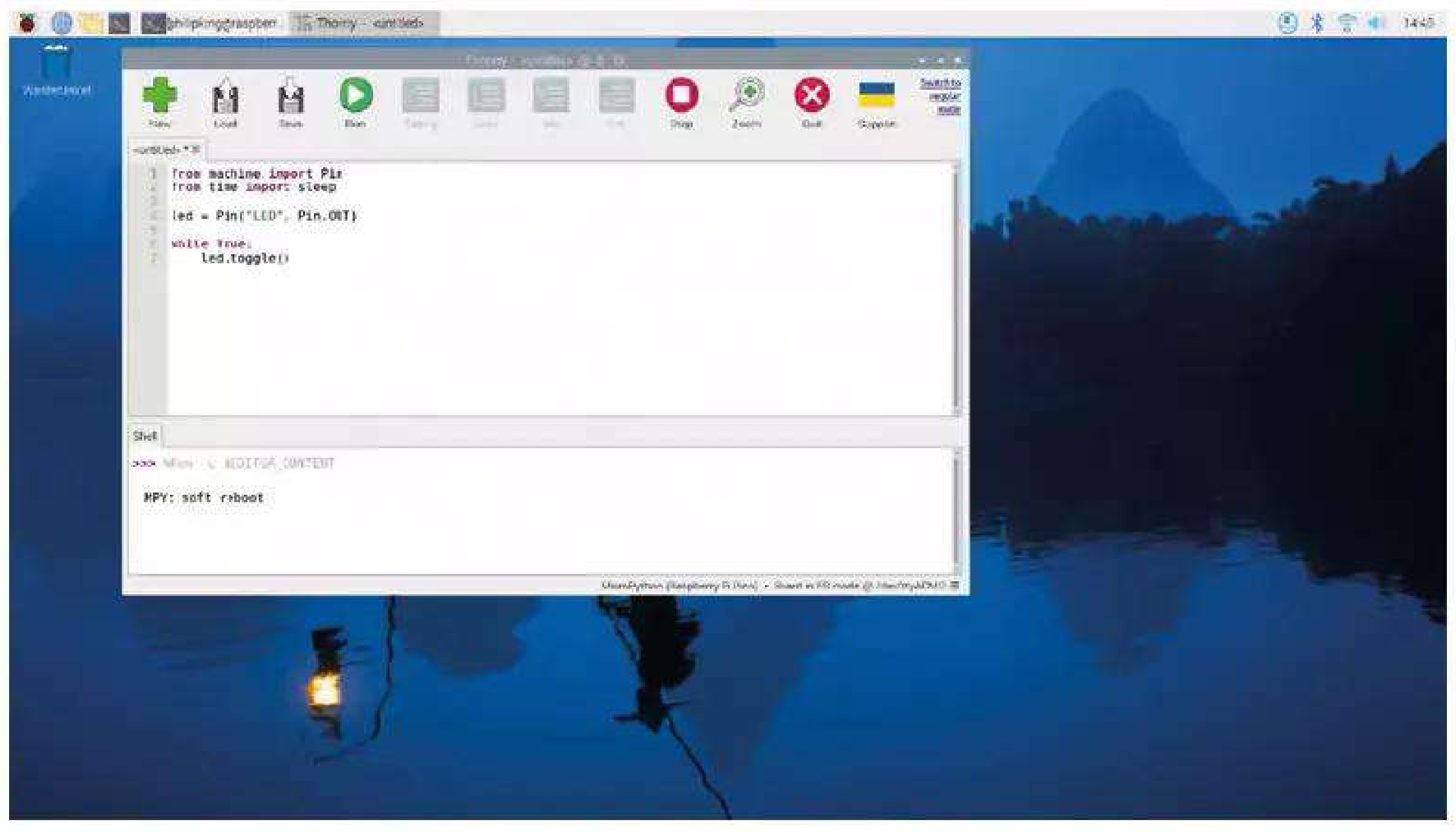
10 To toggle the LED at regular intervals, we'll alter the code. Under line 1, add the following to import the `time` module's sleep class:

```
from time import sleep
```

11 Before `led.toggle`, add a new line to create an infinite loop:

```
while True:
```

Indent `led.toggle` by four spaces (so it becomes part of the loop).



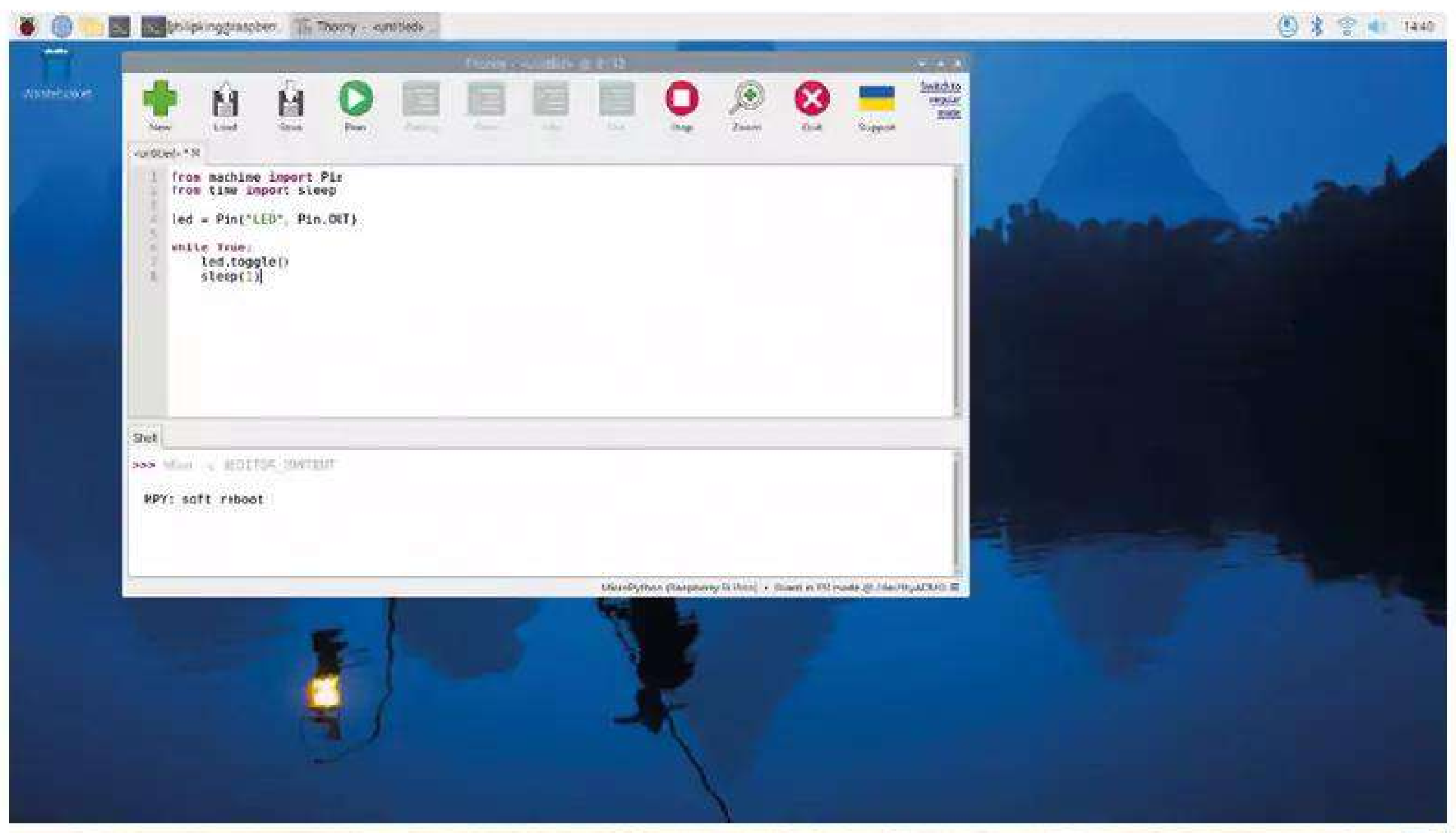
```
from machine import Pin
from time import sleep

led = Pin("LED", Pin.OUT)

while True:
    led.toggle()
```

```
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
```

12 Add another indented line to the loop to add a one-second time delay each time:

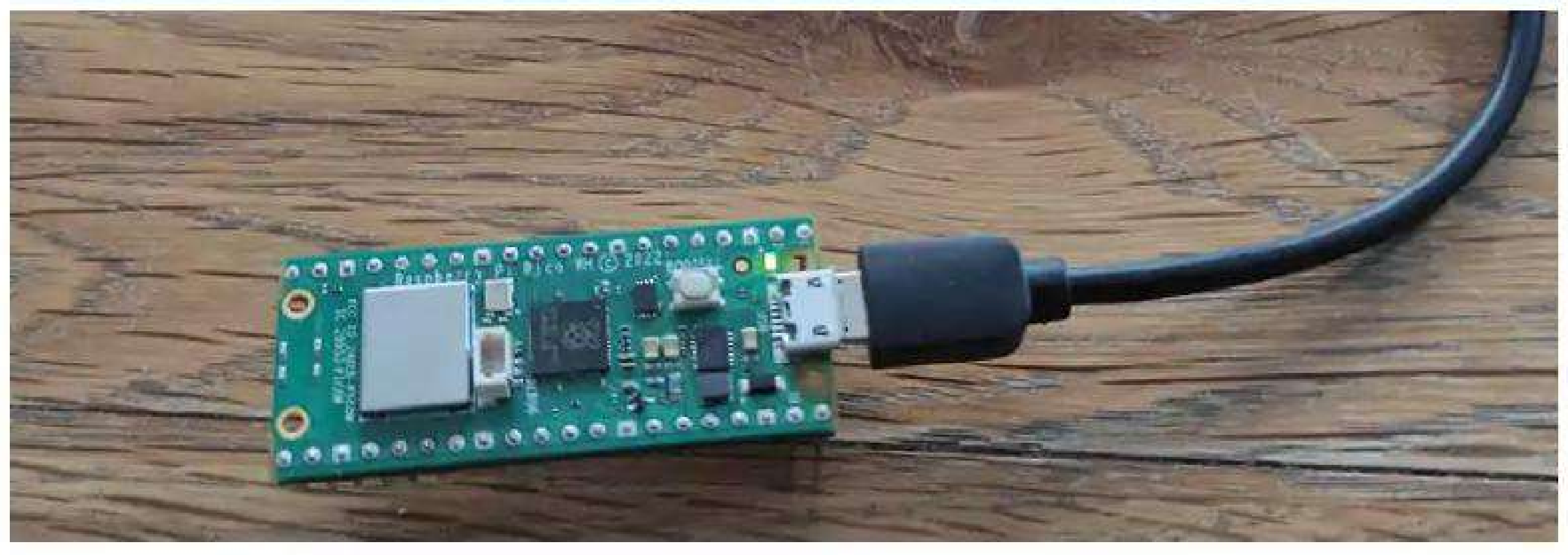
```
sleep(1)
```

Now run the code and the LED should toggle on and off every second.



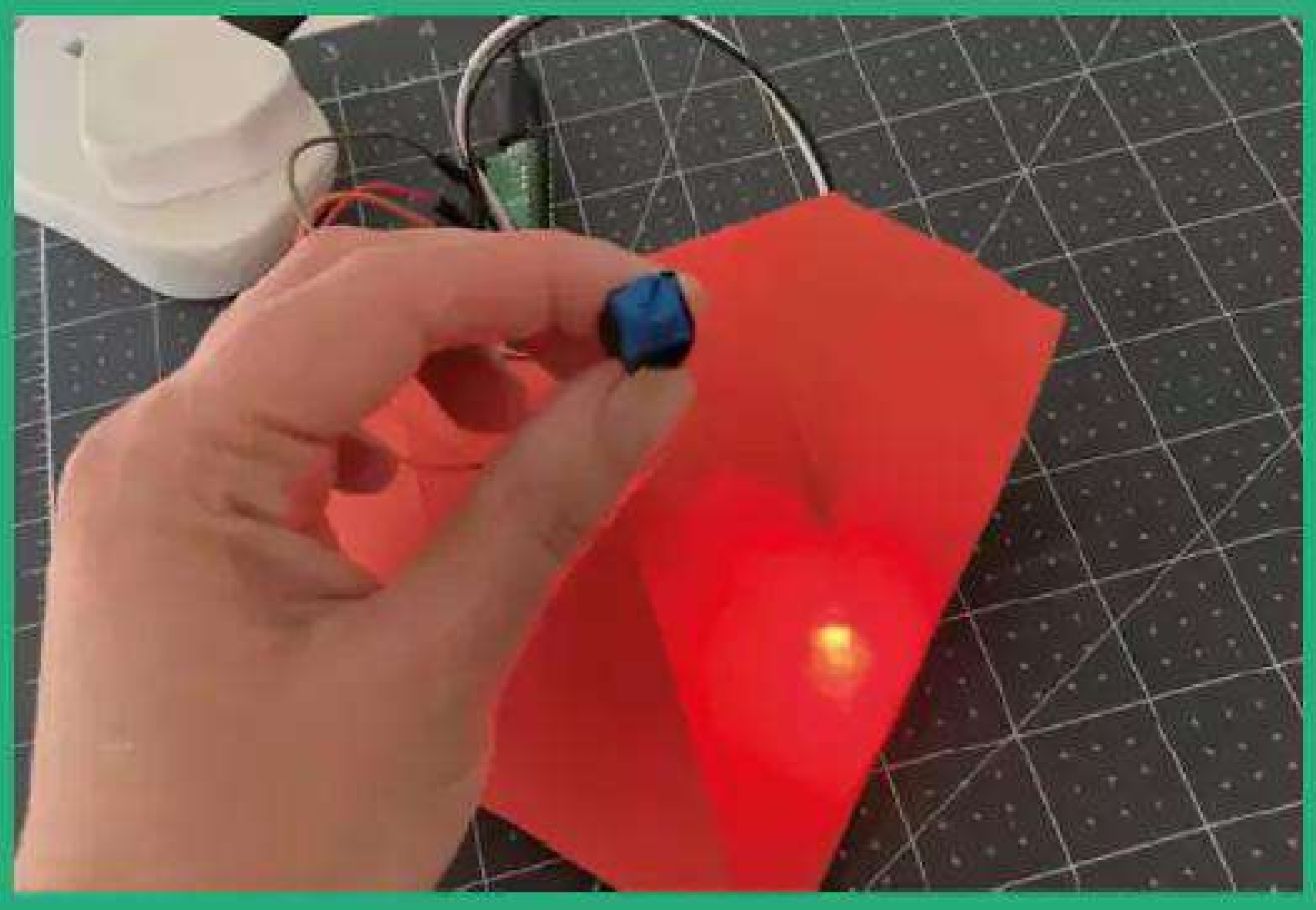
```
from machine import Pin
from time import sleep

led = Pin("LED", Pin.OUT)

while True:
    led.toggle()
    sleep(1)
```

```
>>> %Run -c $EDITOR_CONTENT

MPY: soft reboot
```



Next **Pico projects**

Now you know how to program Pico, try making these beginner projects.

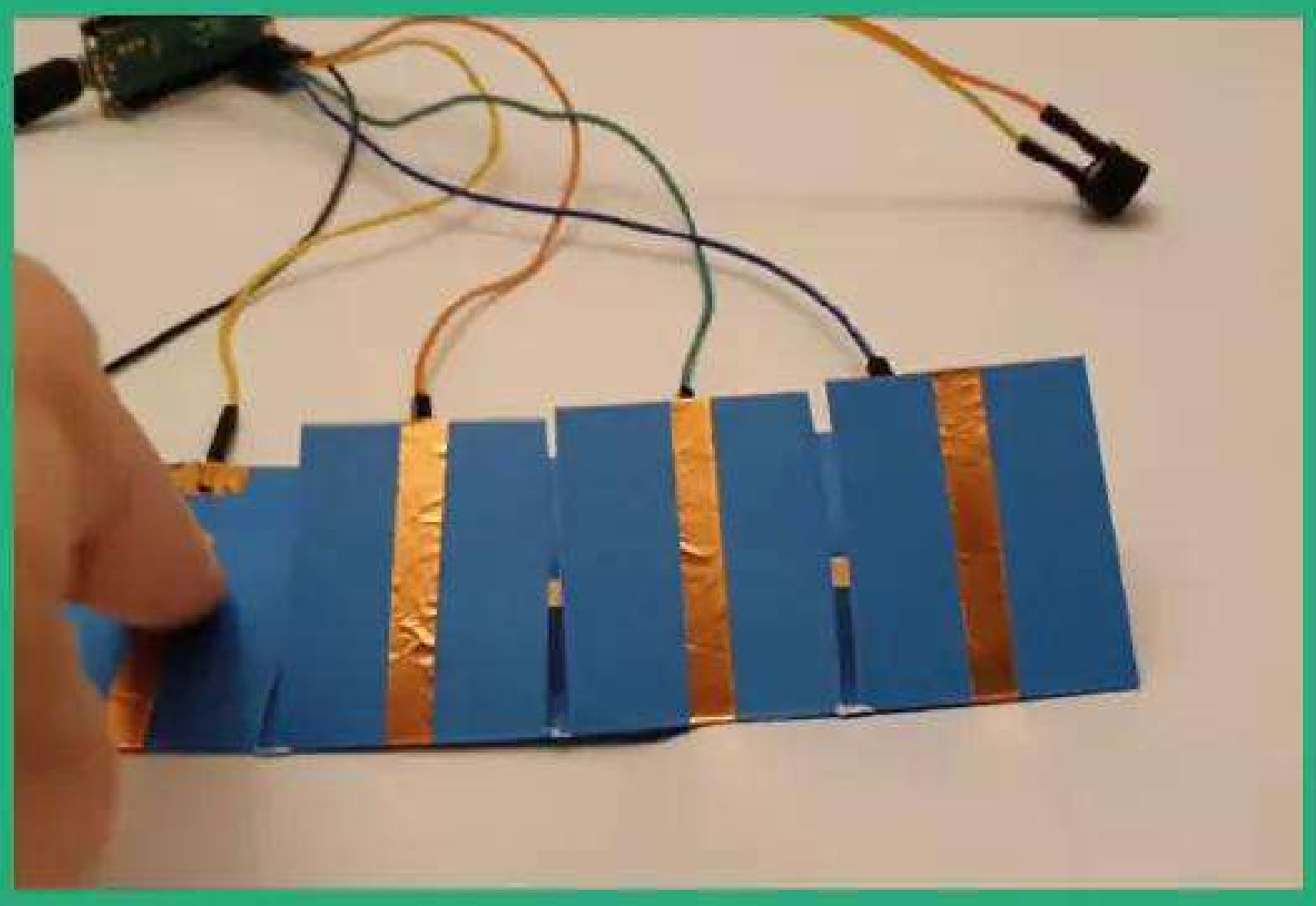


Beating Heart

Place an LED inside a papercraft heart and use a potentiometer to control how often it blinks (the heart rate).
magpi.cc/picoheart

Sensory Gadget

Make an interactive toy such as a Picosaber or Digital Candle with multiple different input types and outputs.
magpi.cc/picogadget



Sound Machine

By connecting one or more buzzers and controls, Pico can be used to play sounds, or used as an alarm.
magpi.cc/picosound

Passive Cooling Open CNC Case

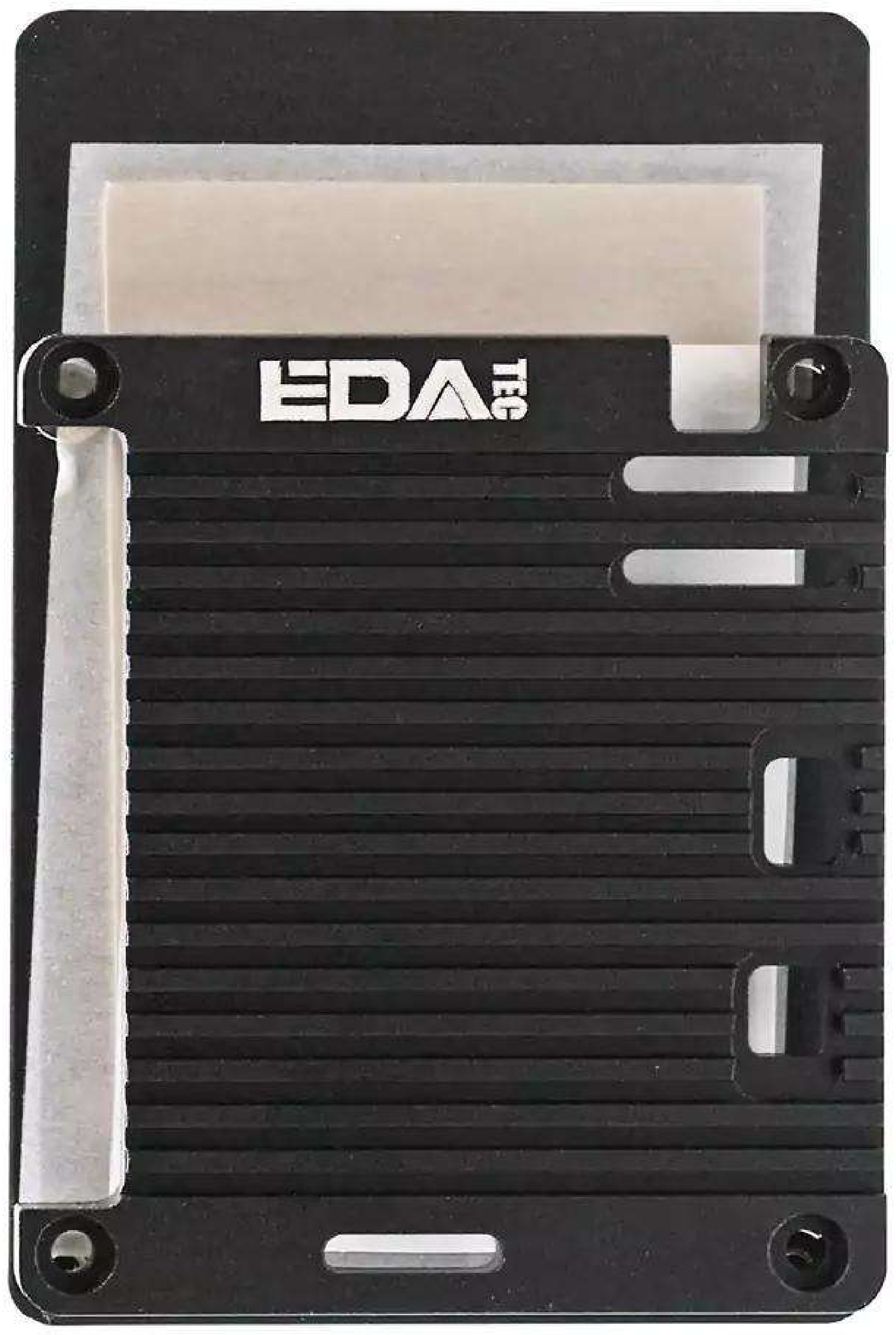
▶ EDATEC ▶ magpi.cc/opencnccase ▶ £7 / $8

A Raspberry Pi 5 case with two heat sinks for serious cooling. By **Phil King**

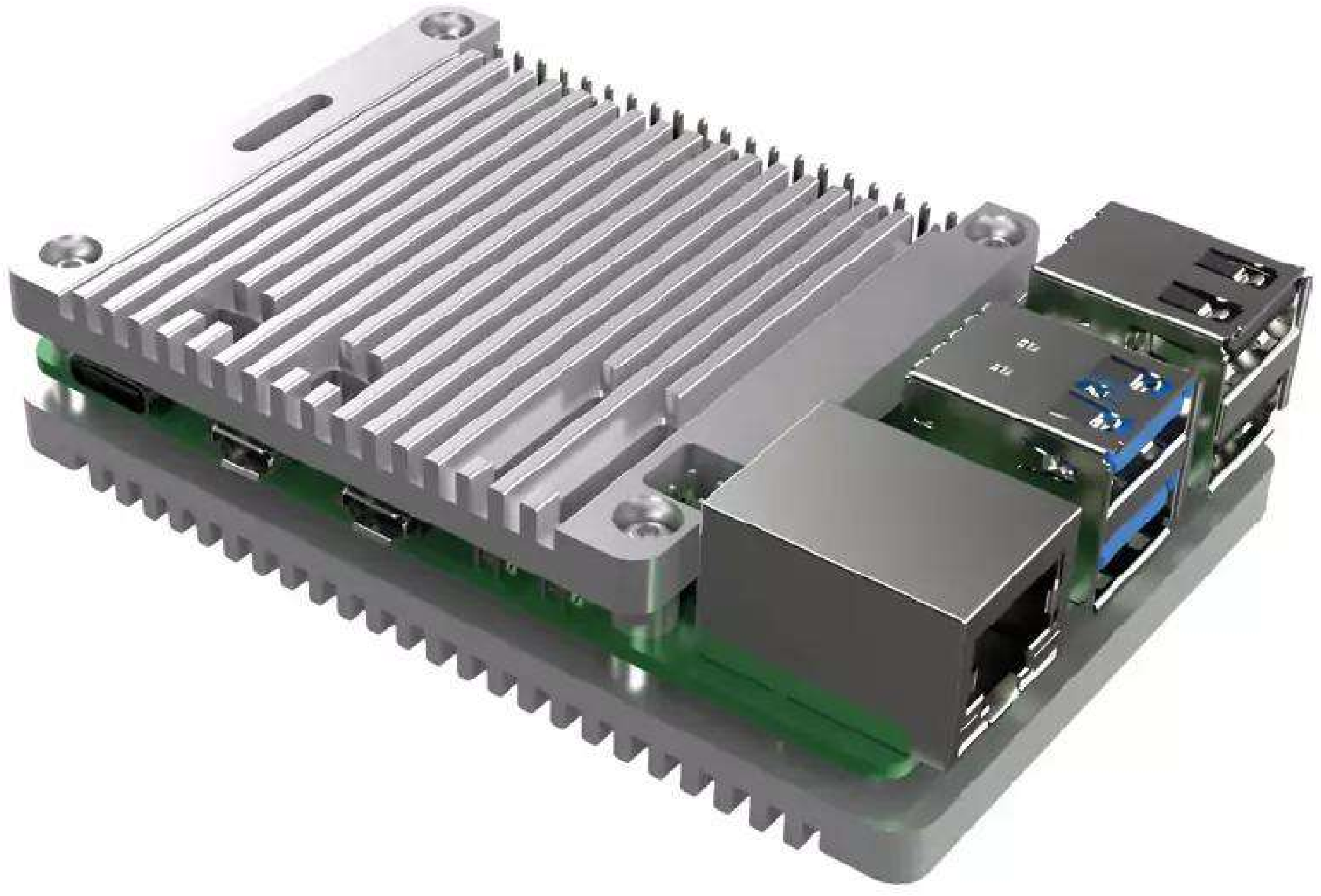
SPECS

COOLING
Two large heatsink panels (top and base) with thermal pads attached

FEATURES
Cutouts/ slots for all Raspberry Pi 5 top-side ports; open sides for the others



▶ The top section features cutouts and slots for full access to all Raspberry Pi 5's connectors



▲ The case comprises two aluminium heatsink parts that sandwich Raspberry Pi 5 between thermal pads

As its name suggests, the Passive Cooling Open CNC Case has an open design, with Raspberry Pi 5 sandwiched between two segments – one on the top and one on the bottom. There are no side pieces, so airflow over the single-board computer is increased – although EDATEC also makes a closed CNC case that also relies on passive cooling.

Both parts are made from aluminium and feel solid and weighty. On one side, each features grooves that have been cut precisely using a CNC machine; not only do these look cool, but they help to dissipate heat.

On the smooth opposite side of the two parts are thermal pads; for the top piece, there are three that stick to the SoC, power management IC, and Wi-Fi and Bluetooth module. The base section is almost totally covered by a single thermal pad that sticks to the underside of Raspberry Pi 5.

Unlimited access

The two case parts are secured in place with long bolts. With no side pieces, access to Raspberry Pi 5's ports is unobstructed. Cutouts in the top part give access to the GPIO pins and PoE header. There are also slots for the two camera/display MIPI ports and the UART and RTC battery connectors, while the fan connector remains uncovered.

> "With no side pieces, access to Raspberry Pi 5's ports is unobstructed"

So, how much cooling does this case provide? A lot! When idle, Raspberry Pi 5 was 10–15°C cooler than without a case; running a stress test, the difference was around 40°C.

Verdict

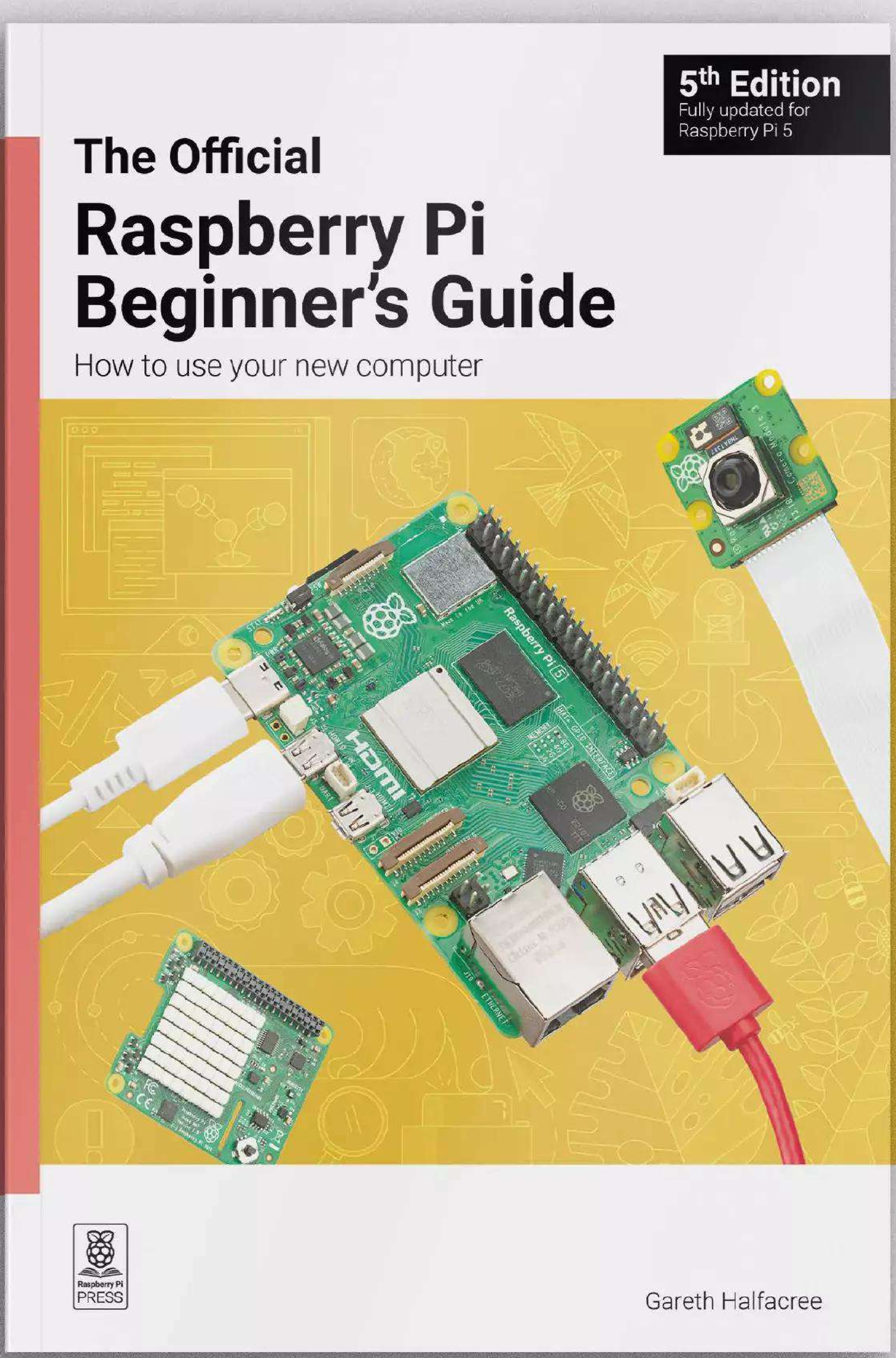
Provides an impressive amount of cooling while giving full access to all Raspberry Pi 5's ports.

9/10

Learn coding

Discover how computers work

Build amazing things!


5th Edition
Fully updated for Raspberry Pi 5
The Official
Raspberry Pi Beginner's Guide
How to use your new computer
Raspberry Pi PRESS
Gareth Halfacree

magpi.cc/beginnersguide

10 Amazing: outdoor projects

Spring has sprung, let's get some new Raspberry Pi projects blooming

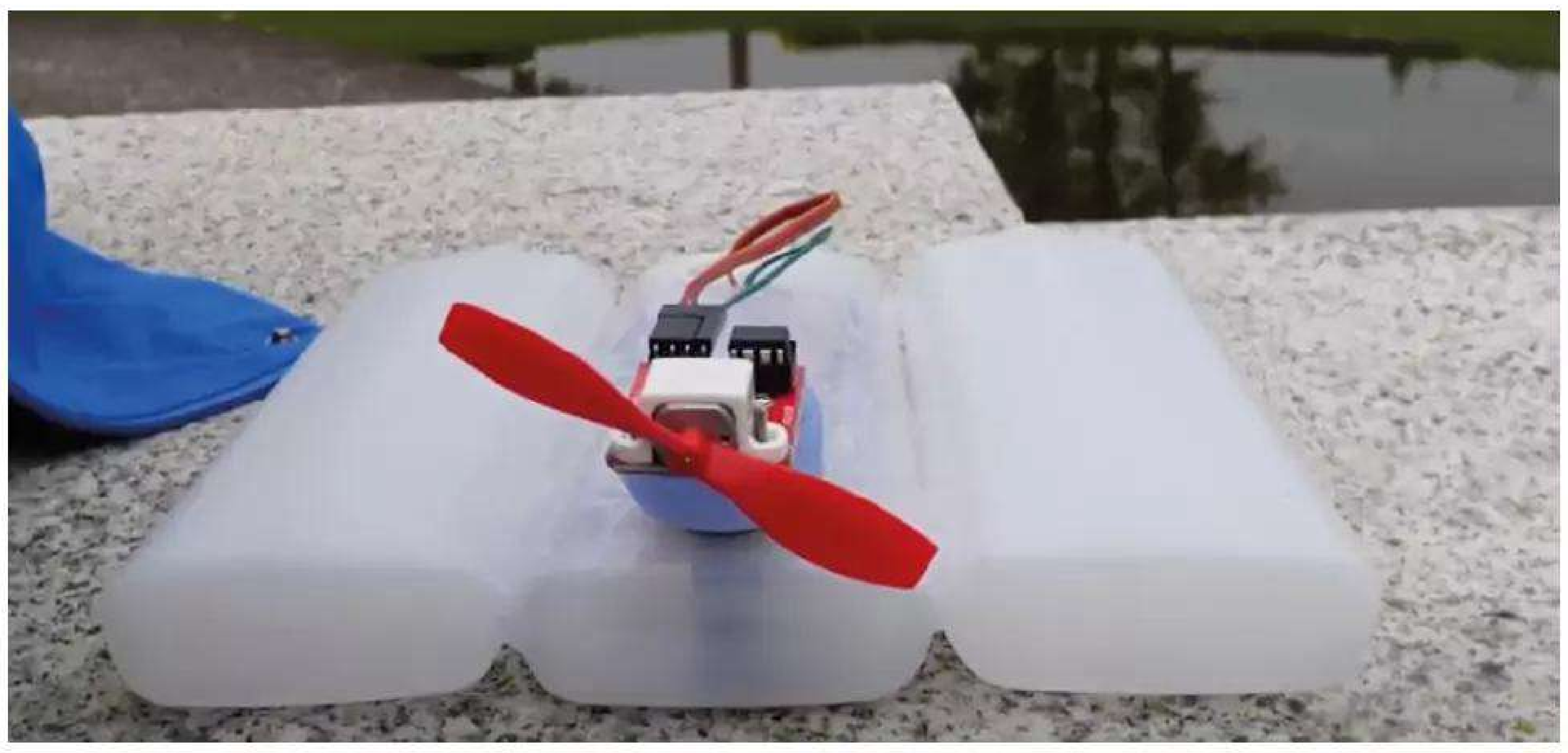
Computing and making may usually be indoor-based hobbies, but that doesn't mean you have to abandon your Raspberry Pi when you go out to touch grass. In fact, taking a Raspberry Pi with you might make the outdoors all that much better – and here are ten of our favourite ways of doing that.

▼ NatureBytes

Spying on critters

What beasties live in your garden? Find out with this (non-invasive!) camera set up that will help you find out if any hedgehogs bless your lawn in the middle of the night

naturebytes.org



▲ Roboat

Pico-powered sailing

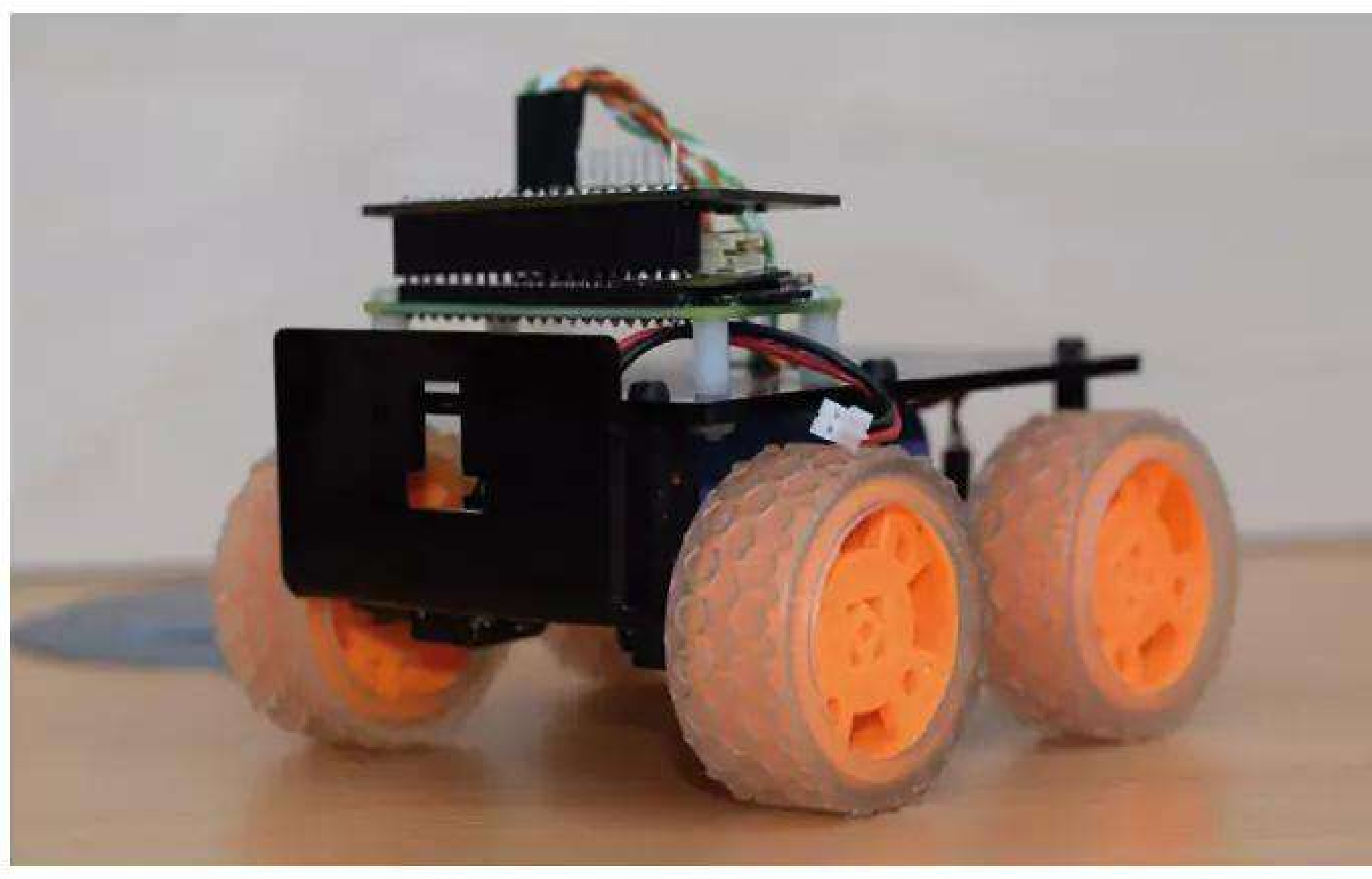
Raspberry Pi Pico is perfect for outdoor projects, especially budget-friendly ones like that use recycled bottles to make a boat hull. It's all controlled from a smartphone app.

magpi.cc/picoroboat

▼ Tiny 4WD robot

Robot and rover

A lot of robots we feature work indoors, however this Tiny 4WD can not only handle outdoor terrain, you can always update the wheels for truly challenging courses – both with a remote control or its own programming.



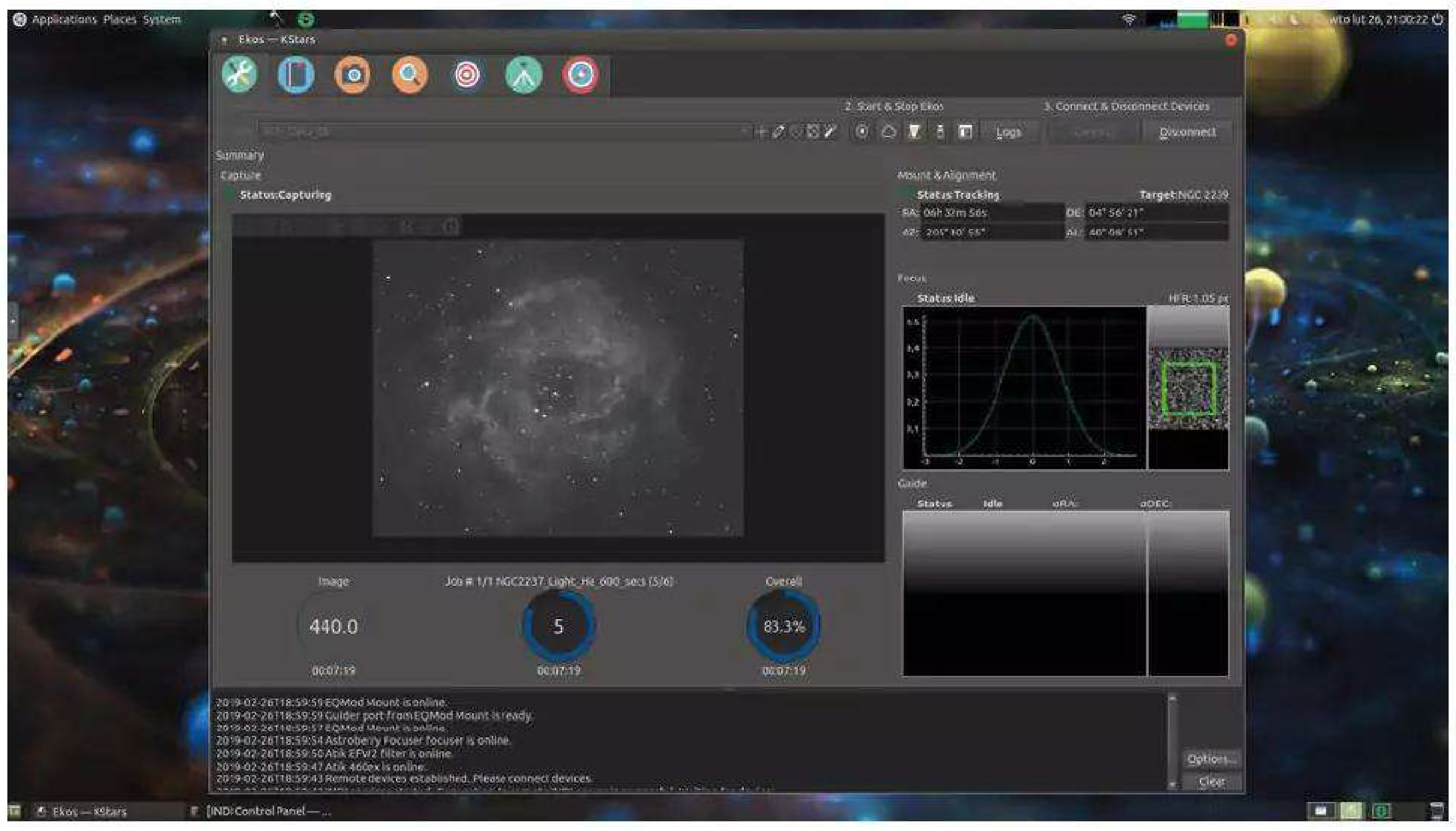
magpi.cc/tiny4wd

▼ Astronomy

See the stars

Pointing your telescope at the right place in the night sky can be tricky – computer control aids in this, and Astroberry on a Raspberry Pi makes it a lot cheaper than other solutions

astroberry.io



Job # 1/1 NGC2237_Light_Ha_600_secs (5/6)

```
2019-02-26T18:59:59 EQMod Mount is online.
2019-02-26T18:59:59 Guider port from EQMod Mount is ready.
2019-02-26T18:59:57 EQMod Mount is online.
2019-02-26T18:59:54 Astroberry Focuser focuser is online.
2019-02-26T18:59:50 Atik EFW2 filter is online.
2019-02-26T18:59:47 Atik 460ex is online.
2019-02-26T18:59:43 Remote devices established. Please connect devices.
```

▲ Astrophotography

Shoot the stars

Using software like Astroberry, along with a Raspberry Pi camera module, you can take stunning time-lapse shots of celestial bodies trillions of miles away to wow yourself and your friends.

magpi.cc/hubblepi

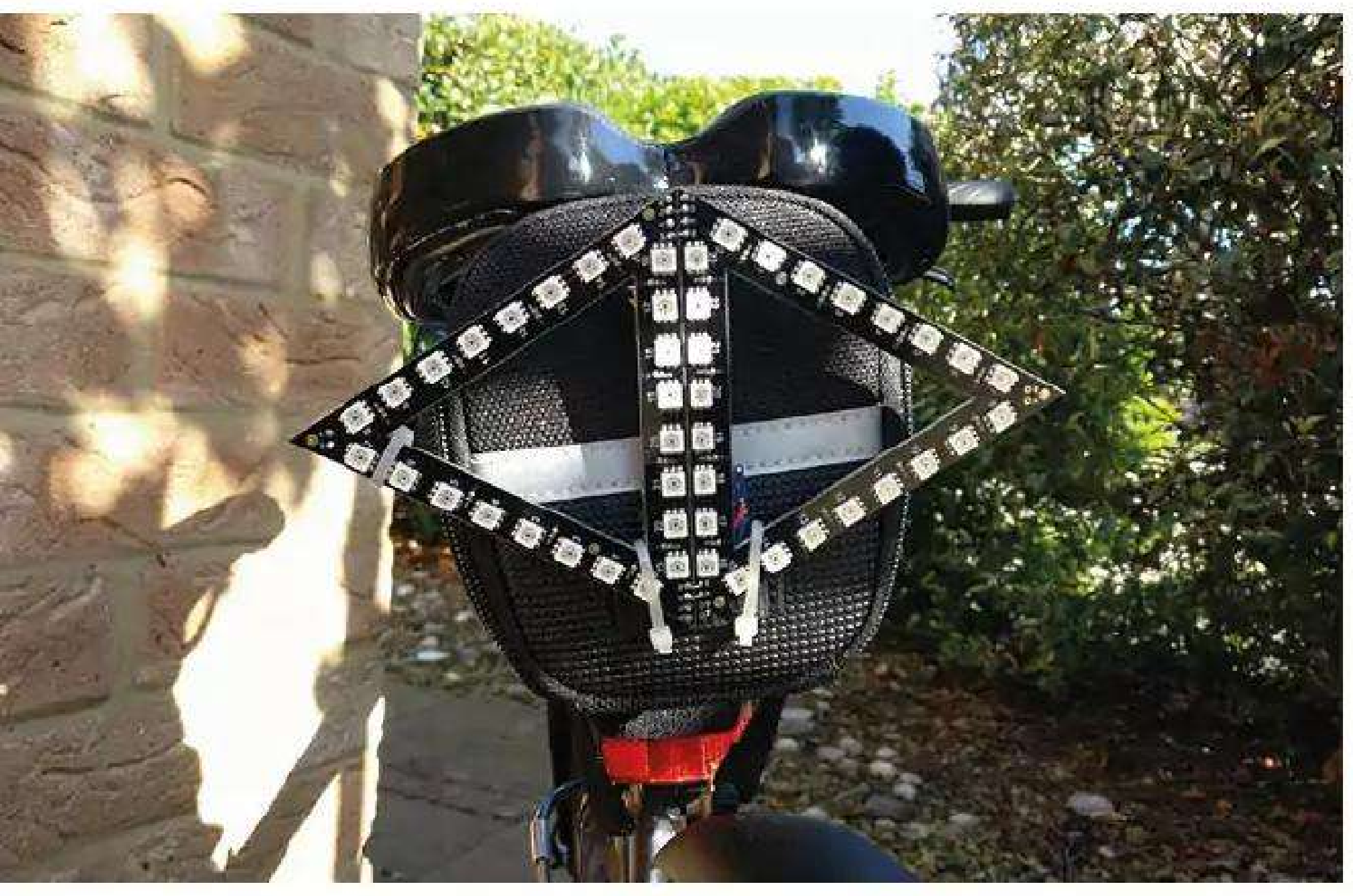
▲ High altitude ballooning

Near-space flights

Going about 40 km straight up, HABs are amazing projects that push the limits of how far making can go - literally. Dave Akerman is the king of Raspberry Pi HABs and has many great tutorials on his website

daveakerman.com

▼ High visibility cycle lights



Indicating easier

Turning some Raspberry Pi-powered LEDs in specific shapes into indicators was no problem for Pete Wood. It started off mobile phone controller but got upgraded with a dedicated indicator switch – very smart!

magpi.cc/bikelights

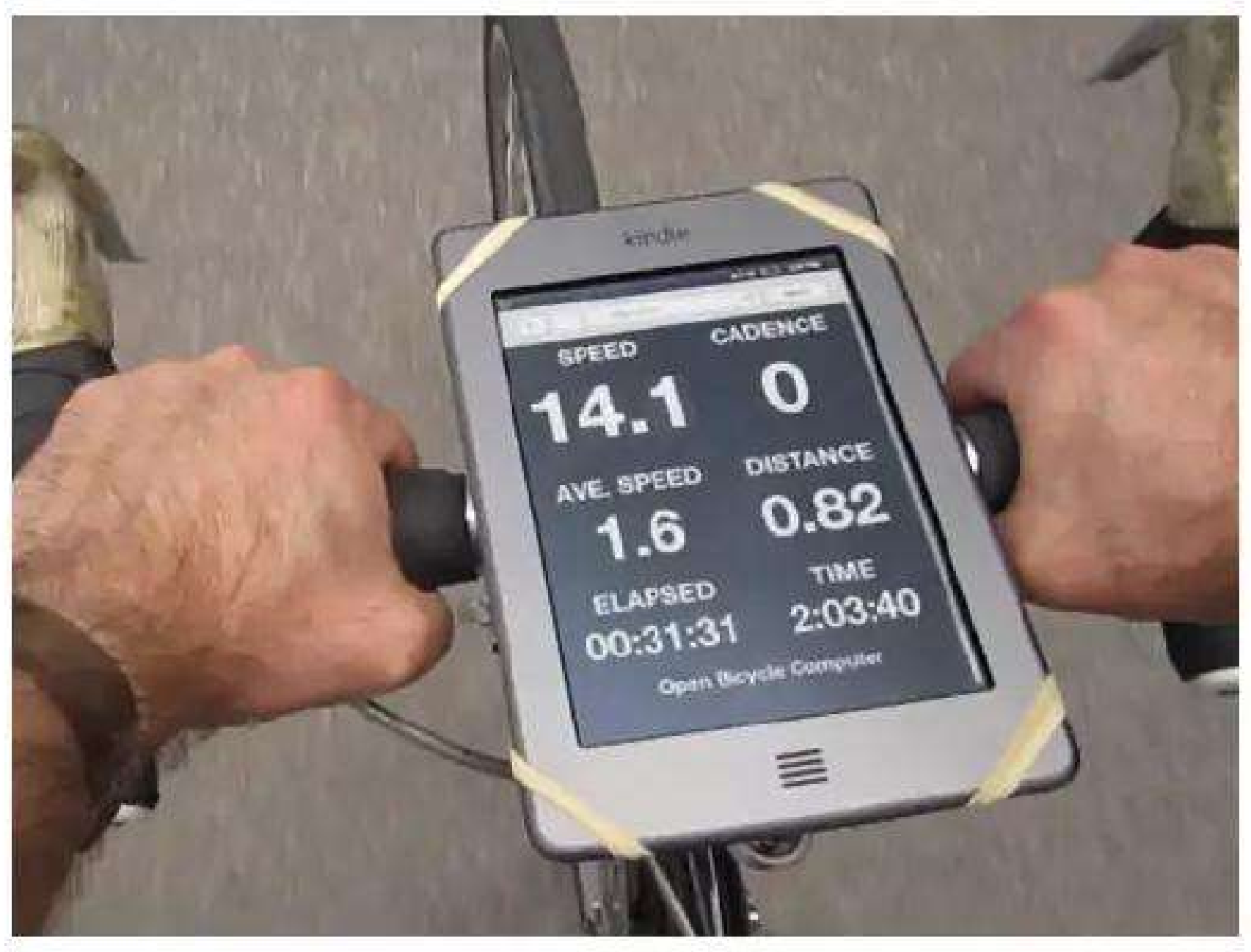
▲ Community Jams

Raspberry Pi Parklife

During COVID, a group of musicians found the only way they could play was outside. One of their members created a sheet music app on Raspberry Pi that all the members could play along to. The jams were able to continue.

magpi.cc/comjams

▼ Build a better bike computer



Cheap and practical

In a quest to make a more readable bike computer, maker David Schneider repurposed a touchscreen Kindle to display important riding info via Raspberry Pi

magpi.cc/bikecomp

▼ MudPi

Treat your garden

This modular, open-source, automated garden system will help keep your garden perfectly irrigated – perfect for the summer just around the corner and the numerous BBQ parties you'll have

magpi.cc/mudpi

Master Bash with Raspberry Pi

Dive into the command line with these top resources.
By **Lucy Hattersley**

Linux Command Line and Shell Scripting Bible

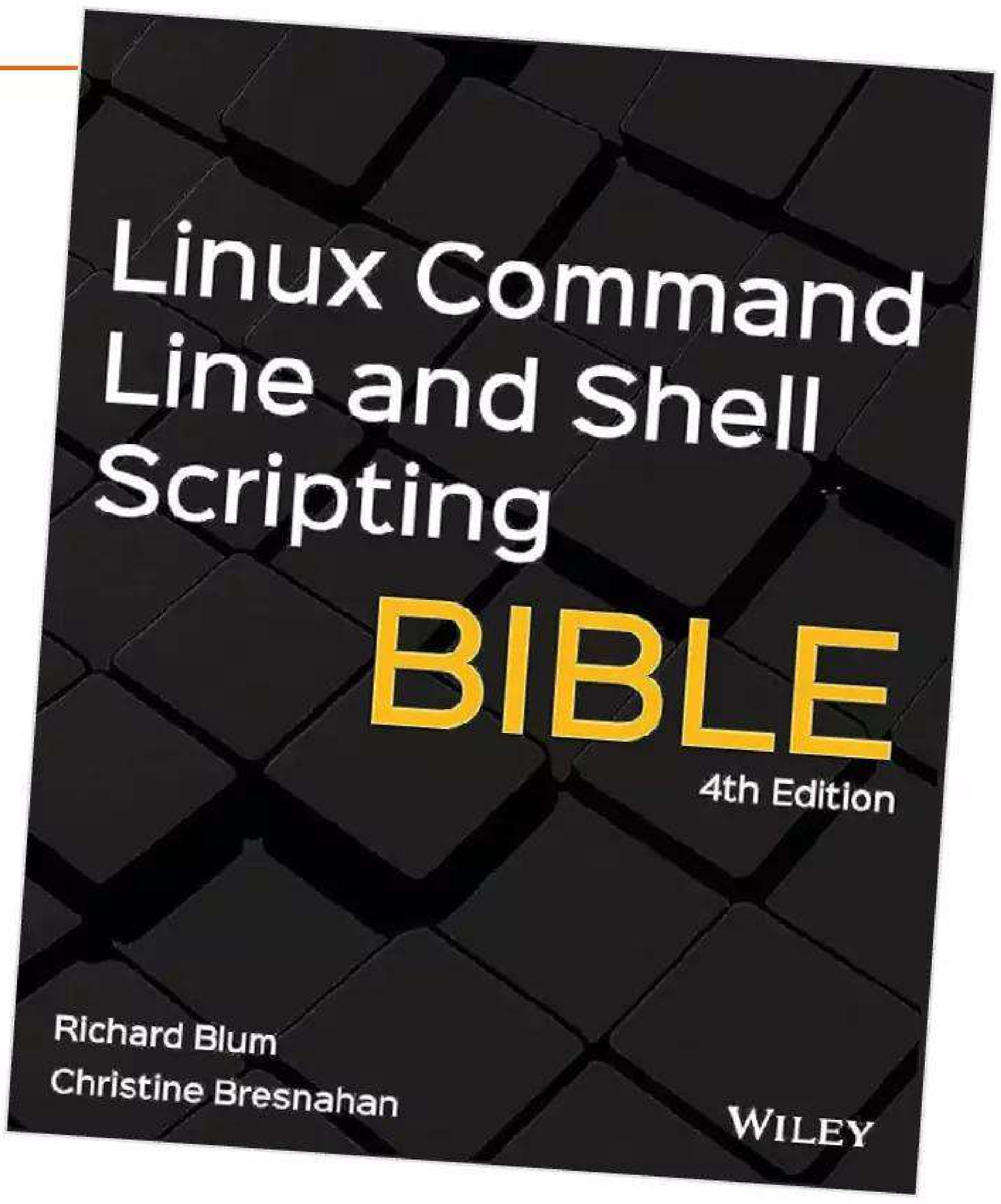
AUTHOR
Richard Blum and Christine Bresnahan

Price:
£40/$40

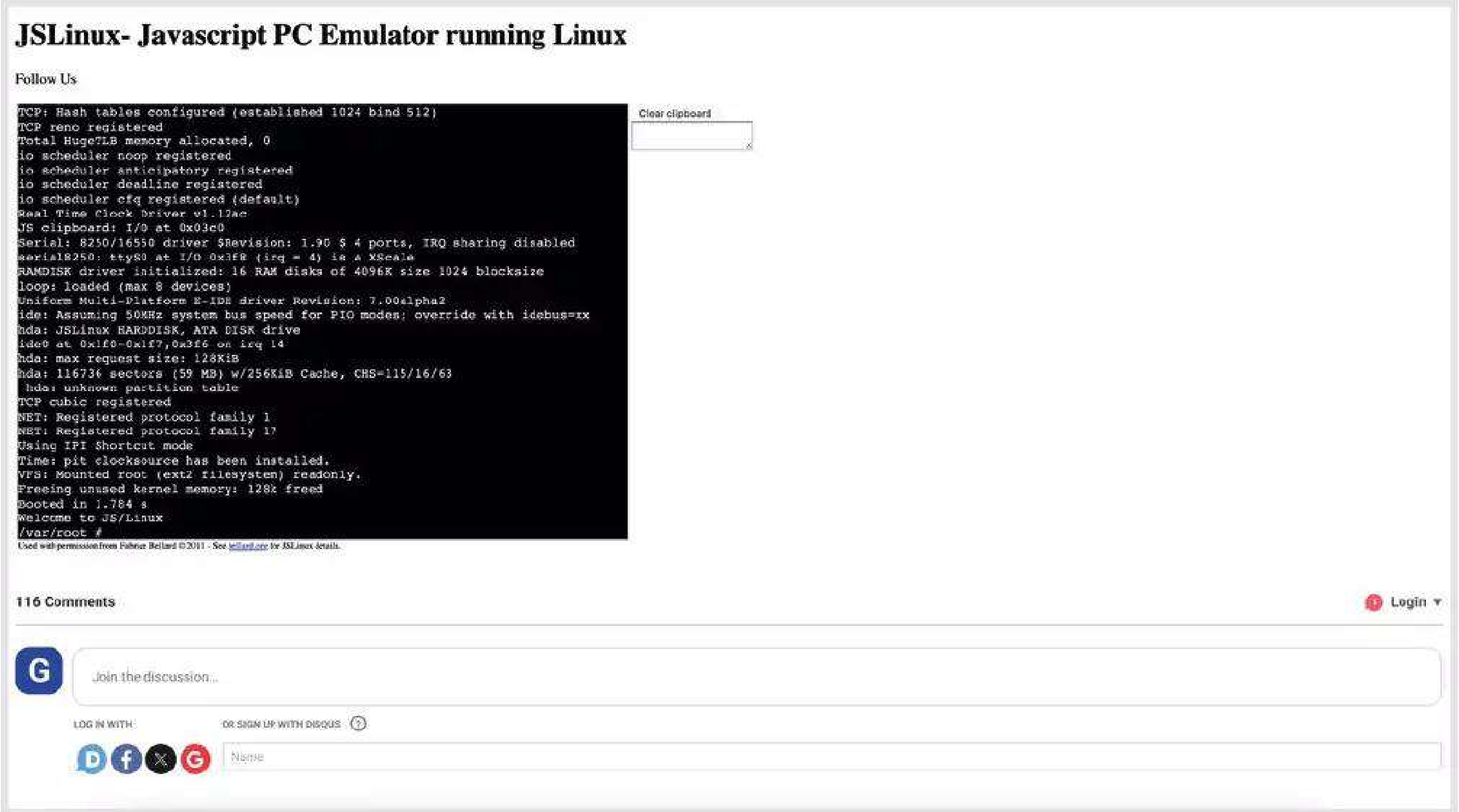
magpi.cc/bashscriptyt

For those looking for a serious dive deep into the command line and shell scripting, the *Linux Command Line and Shell Scripting Bible* is a must-have.

This comprehensive guide is packed with detailed explanations, practical examples, and hands-on exercises that cover everything from basic commands to advanced scripting. Whether you're a beginner aiming to get acquainted with the command line or an experienced user looking to enhance your scripting skills, this book provides valuable insights and tips that can be immediately applied to improve your efficiency and productivity in the Linux environment.



Bash web resources


JSLinux- Javascript PC Emulator running Linux

Follow Us

```
TCP: Hash tables configured (established 1024 bind 512)
TCP reno registered
Total HugeTLB memory allocated, 0
io scheduler noop registered
io scheduler anticipatory registered
io scheduler deadline registered
io scheduler cfq registered (default)
Real Time Clock Driver v1.12ac
JS clipboard: I/O at 0x03c0
Serial: 8250/16550 driver $Revision: 1.90 $ 4 ports, IRQ sharing disabled
serial8250: ttyS0 at I/O 0x3f8 (irq = 4) is a XScale
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
loop: loaded (max 8 devices)
Uniform Multi-Platform E-IDE driver Revision: 7.00alpha2
ide: Assuming 50MHz system bus speed for PIO modes; override with idebus=xx
hda: JSLinux HARDDISK, ATA DISK drive
ide0 at 0x1f0-0x1f7,0x3f6 on irq 14
hda: max request size: 128KiB
hda: 116736 sectors (59 MB) w/256KiB Cache, CHS=115/16/63
 hda: unknown partition table
TCP cubic registered
NET: Registered protocol family 1
NET: Registered protocol family 17
Using IPI Shortcut mode
Time: pit clocksource has been installed.
VFS: Mounted root (ext2 filesystem) readonly.
Freeing unused kernel memory: 128k freed
Booted in 1.784 s
Welcome to JS/Linux
/var/root #
```

Used with permission from Fabrice Bellard © 2011 - See bellard.org for JSLinux details.

116 Comments

Three handy websites to keep in your bookmarks

JSLINUX

Use this JavaScript Linux emulator to experiment with shell commands safely.
▶ jslinux.org

BASH REFERENCE GUIDE

Keep this reference guide to hand to get information on any commands.
▶ magpi.cc/bashman

BASH CHEAT SHEET

Bookmark this cheat sheet to quickly scan for commands you need to know.
▶ webminal.org

Learn Shell

AUTHOR

Learnshell

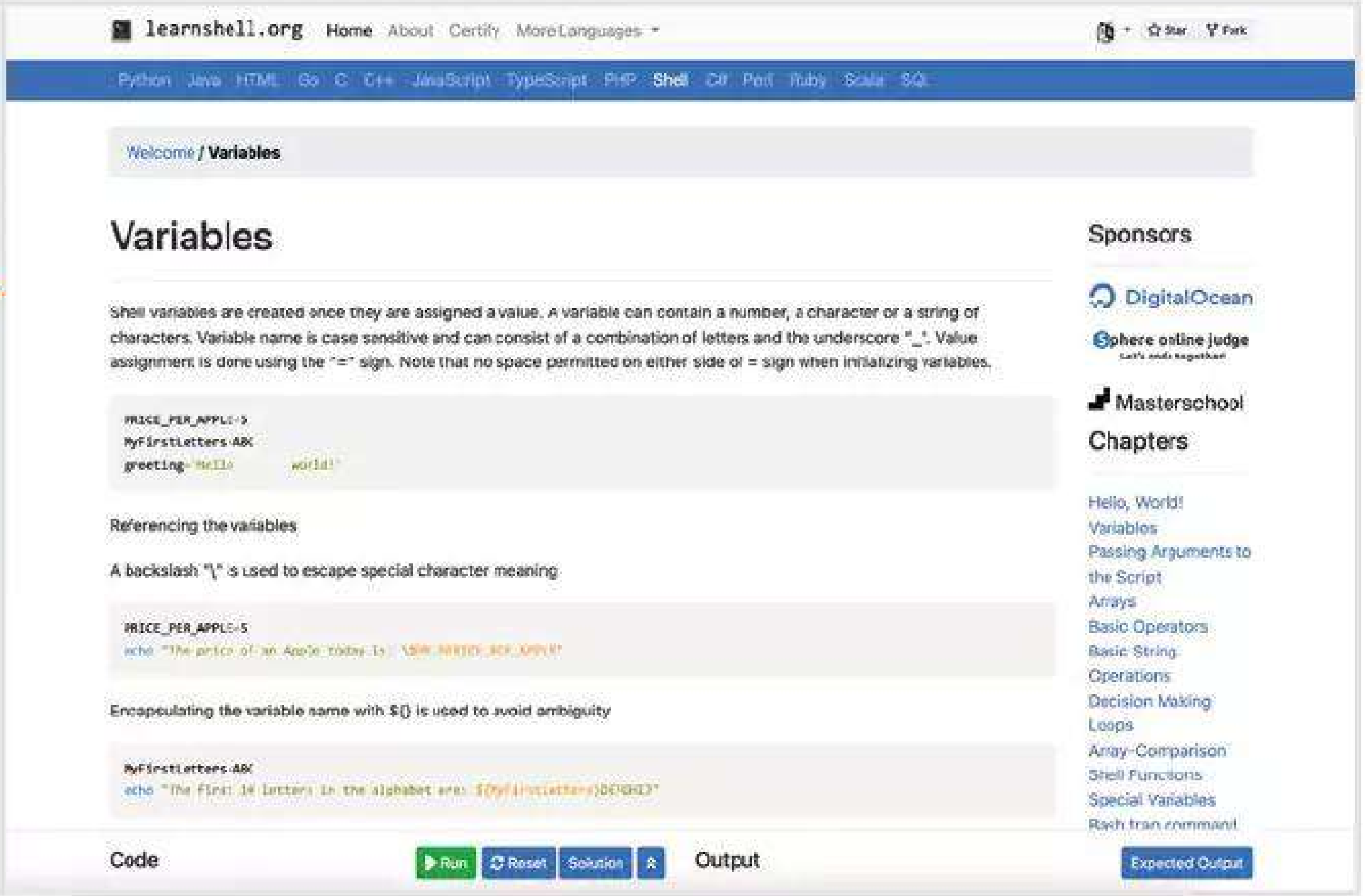
Price:
Free

learnshell.org

Learn Shell is an interactive web tutorial for those new to shell scripting or seeking to strengthen their skills. You write shell scripts directly into a browser, and it tells you if the output is correct or not.

A series of lessons progress from basic to advanced topics, and users will develop a solid understanding of shell commands and scripting concepts at their own pace. Each lesson focuses on practical applications and examples to show the power and flexibility of shell scripting.

It's particularly helpful for those who prefer an interactive approach to reading through dense documentation or textbooks.



Variables

Shell variables are created once they are assigned a value. A variable can contain a number, a character or a string of characters. Variable name is case sensitive and can consist of a combination of letters and the underscore "_". Value assignment is done using the "=" sign. Note that no space permitted on either side of = sign when initializing variables.

```
PRICE_PER_APPLE=5
MyFirstLetters=ABC
greeting='Hello        world!'
```

Referencing the variables

A backslash "\" is used to escape special character meaning

```
PRICE_PER_APPLE=5
echo "The price of an Apple today is: \$HK $PRICE_PER_APPLE"
```

Encapsulating the variable name with ${} is used to avoid ambiguity

```
MyFirstLetters=ABC
echo "The first 10 letters in the alphabet are: ${MyFirstLetters}DEFGHIJ"
```

Bash Beginner Tutorials

AUTHOR

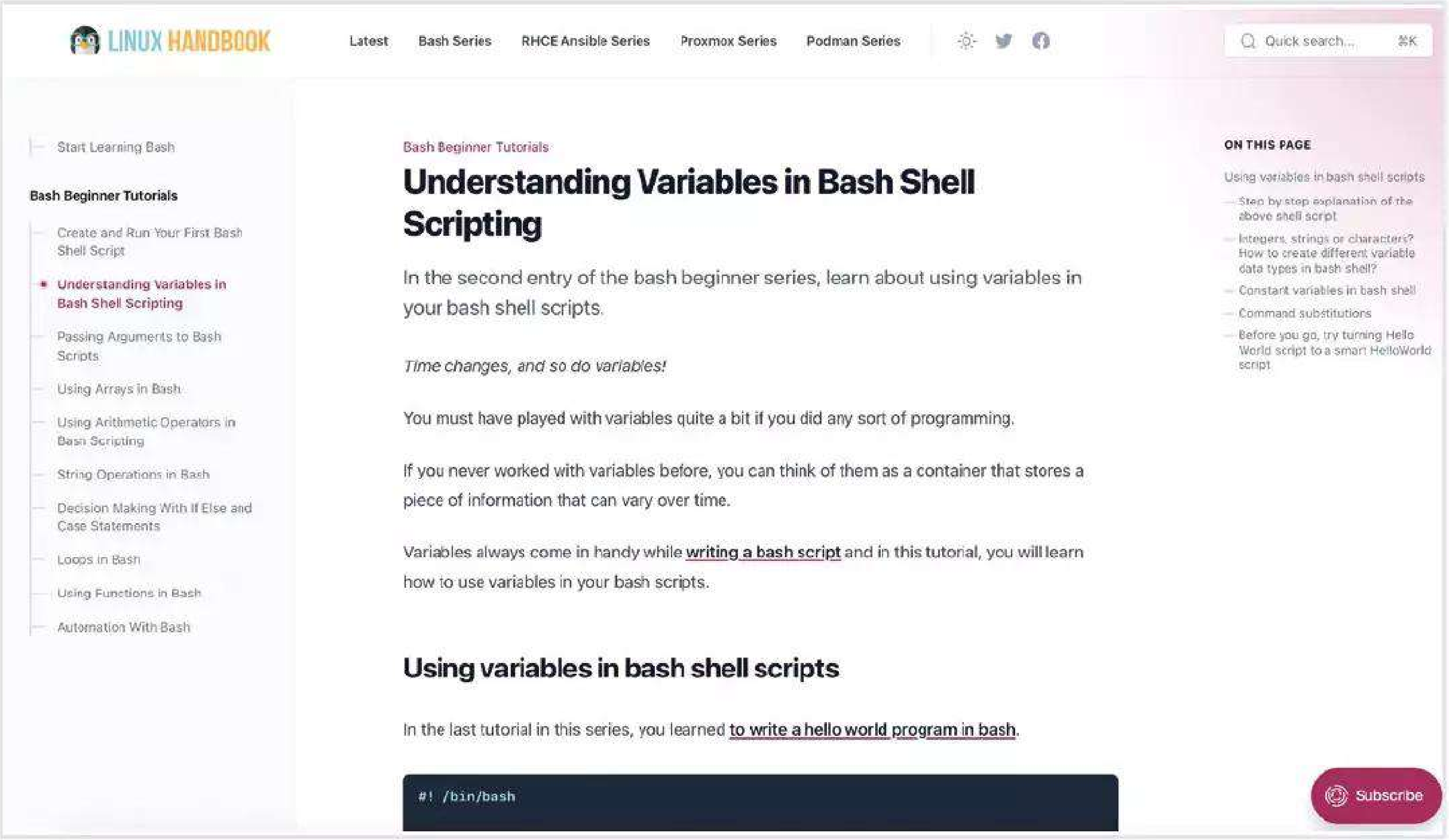
Linux Handbook

Price:
Free

linuxhandbook.com/bash

Linux Handbook is a good online tutorial series for those beginning their journey with the bash shell. The series introduces essential bash commands, scripting fundamentals, and automation.

Each tutorial is accompanied by exercises you can run in the terminal. It's ideal for beginners seeking a comprehensive and approachable guide to Bash, offering a strong foundation for more advanced study.


LINUX HANDBOOK
Latest
Bash Series
RHCE Ansible Series
Proxmox Series
Podman Series
Quick search... ⌘K
Start Learning Bash
Bash Beginner Tutorials
Create and Run Your First Bash Shell Script
Understanding Variables in Bash Shell Scripting
Passing Arguments to Bash Scripts
Using Arrays in Bash
Using Arithmetic Operators in Bash Scripting
String Operations in Bash
Decision Making With If Else and Case Statements
Loops in Bash
Using Functions in Bash
Automation With Bash
ON THIS PAGE
Using variables in bash shell scripts
Step by step explanation of the above shell script
Integers, strings or characters? How to create different variable data types in bash shell?
Constant variables in bash shell
Command substitutions
Before you go, try turning Hello World script to a smart HelloWorld script
Subscribe

Bash Beginner Tutorials

Understanding Variables in Bash Shell Scripting

In the second entry of the bash beginner series, learn about using variables in your bash shell scripts.

Time changes, and so do variables!

You must have played with variables quite a bit if you did any sort of programming.

If you never worked with variables before, you can think of them as a container that stores a piece of information that can vary over time.

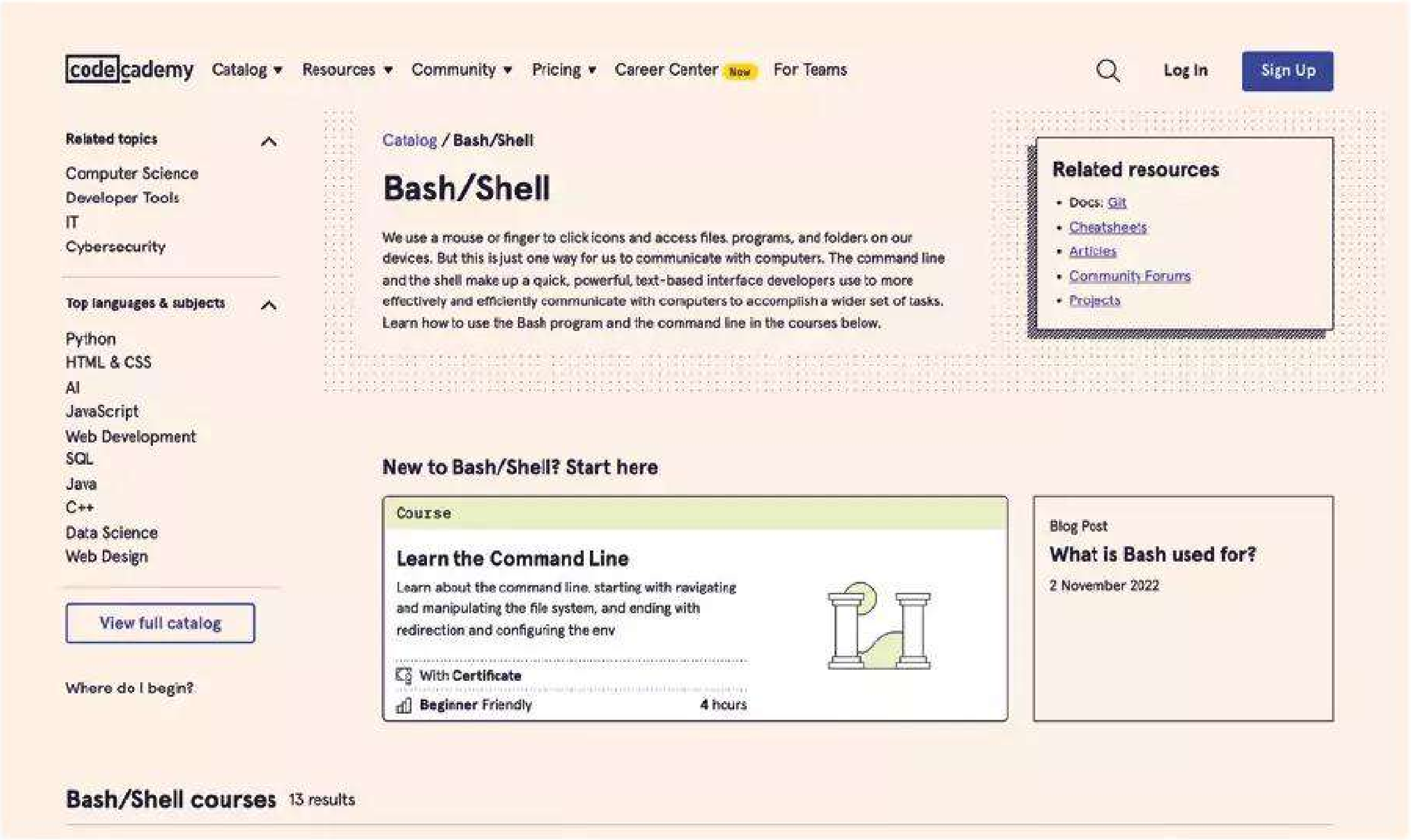
Variables always come in handy while writing a bash script and in this tutorial, you will learn how to use variables in your bash scripts.

Using variables in bash shell scripts

In the last tutorial in this series, you learned to write a hello world program in bash.

```
#! /bin/bash
```

Follow these courses



Bash/Shell

We use a mouse or finger to click icons and access files, programs, and folders on our devices. But this is just one way for us to communicate with computers. The command line and the shell make up a quick, powerful, text-based interface developers use to more effectively and efficiently communicate with computers to accomplish a wider set of tasks. Learn how to use the Bash program and the command line in the courses below.

New to Bash/Shell? Start here

Learn the Command Line

Learn about the command line, starting with navigating and manipulating the file system, and ending with redirection and configuring the env

What is Bash used for?

2 November 2022

Bash/Shell courses 13 results

These in-depth courses explain the concepts of Bash and terminalbookmarks

BASH SCRIPTING

Free Code Camp has this video series on Bash Scripting Tutorial for Beginners if you prefer to learn from a video tutorial.

▶ magpi.cc/bashscriptyt

BASH/SHELL

Codecademy has a variety of courses for Bash with cheatsheets, articles and forums

▶ magpi.cc/codecademybash

UDEMY BASH

Udemy has a variety of Bash courses for you to take part in. The Complete Bash/Shell Developer course is a comprehensive offering.

▶ magpi.cc/udemybash

Sara Parodi

Meet *The MagPi*'s new designer, whose work you've been seeing on Raspberry Pi products for years

> Name **Sara Parodi** | > Occupation **Graphic designer**
> Community role **The MagPi designer** | > URL **raspberrypi.com**

If you read last issue's Final Word page, or checked the credits in our flannel panel, you'll know that we have moved the design for *The MagPi* in-house. While the whole design team does little bits and pieces here and there, Sara Parodi has been doing most of the work to turn our words into something that looks pretty.

"Growing up, I went to art school in Italy," Sara says. "It felt like the right path for me to undertake, and through the years I pushed myself to try different things; I would find myself at events drawing on walls or floors, other times I would be at home testing lino printing using a DIY device made from an olive press. I'd even try painting on T-shirts or home-printing personalised stickers to stick everywhere. At the time I couldn't figure out a specific definition of what I wanted to do, but nowadays I feel that the best way to explain it is that I love to work as a visual communicator. So here I am."

YOUR OFFICIAL RASPBERRY PI MAGAZINE
The MagPi
COMPLETE RASPBERRY PI BUYER'S GUIDE
OVERCLOCK & BENCHMARK RASPBERRY PI 5
MAKE BUILD HACK CREATE
HackSpace
TECHNOLOGY IN YOUR HANDS
Raspberry Pi 5 PROJECTS
3D PRINTING ARDUINO NEURAL NETWORKS
3D Evolution
PRUSA'S NEWEST PRINTER ON TEST
PICO CHATGPT CATAPULT SCOPE

▼ The team at Raspberry Pi have been working on our covers for a while now, and have helped make some truly excellent ones

How did you join Raspberry Pi?
At the end of 2021 I was submitting my final major project for an illustration MA at Falmouth University. In January I was ready to get back into the creative industry, so I began looking for job vacancies when I came across to a catchy one on LinkedIn. In a series of paragraphs, I saw described what I enjoy doing, and I applied for it straight away. The interview was great and I remember coming back home very enthusiastic about it.

What Raspberry Pi design stuff have you worked on?
I'm lucky enough to be in a place where projects vary, and the nice part is that depending on the scale of them I might be working on my own or with the skillset of my colleagues. So far I've worked on a variety of projects of different scales,

▲ Raspberry Pi prides itself on simple yet beautiful packaging – we especially love the new Touch Display designs which Sara worked on

such as packaging design, visual communication for events and related merch, design layout for case studies, books, flyers, brochures and now magazines!

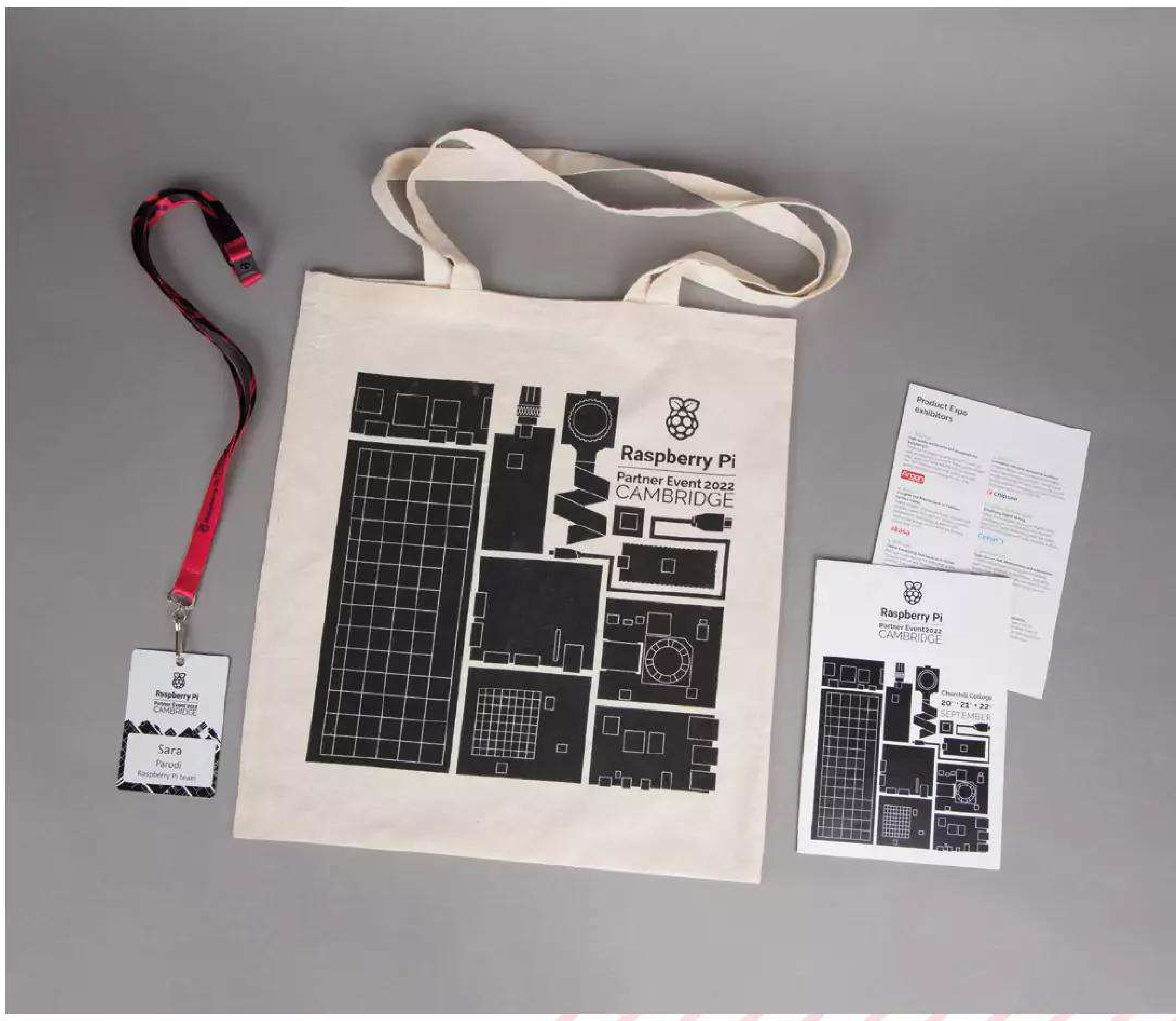
> “ I love creating bodies of work that mix illustration with fine art ”

Have you made anything with a Raspberry Pi, or have any plans to?
I'd never done any coding before joining Raspberry Pi, so I'm still in a phase of learning while watching [my partner] doing some small home projects. Last year I participated in a couple of workshops on using Pico on a breadboard, and learned how to turn on some LEDs!

In regards to future plans… I have an idea for creating something that involves my artwork, like a sort of flipping book installation, or a projected animated GIF, but I need to define the idea first and understand how to use Raspberry Pi with it.

What other hobbies do you have?
I love creating bodies of work that mix illustration with fine art, along with testing new materials and techniques. I usually do this at the studio space I share with other creatives in the heart of Cambridge. I enjoy going there to meet them for a coffee or to chat about what they're up to. Sometimes we organise open studios or small exhibitions next door.

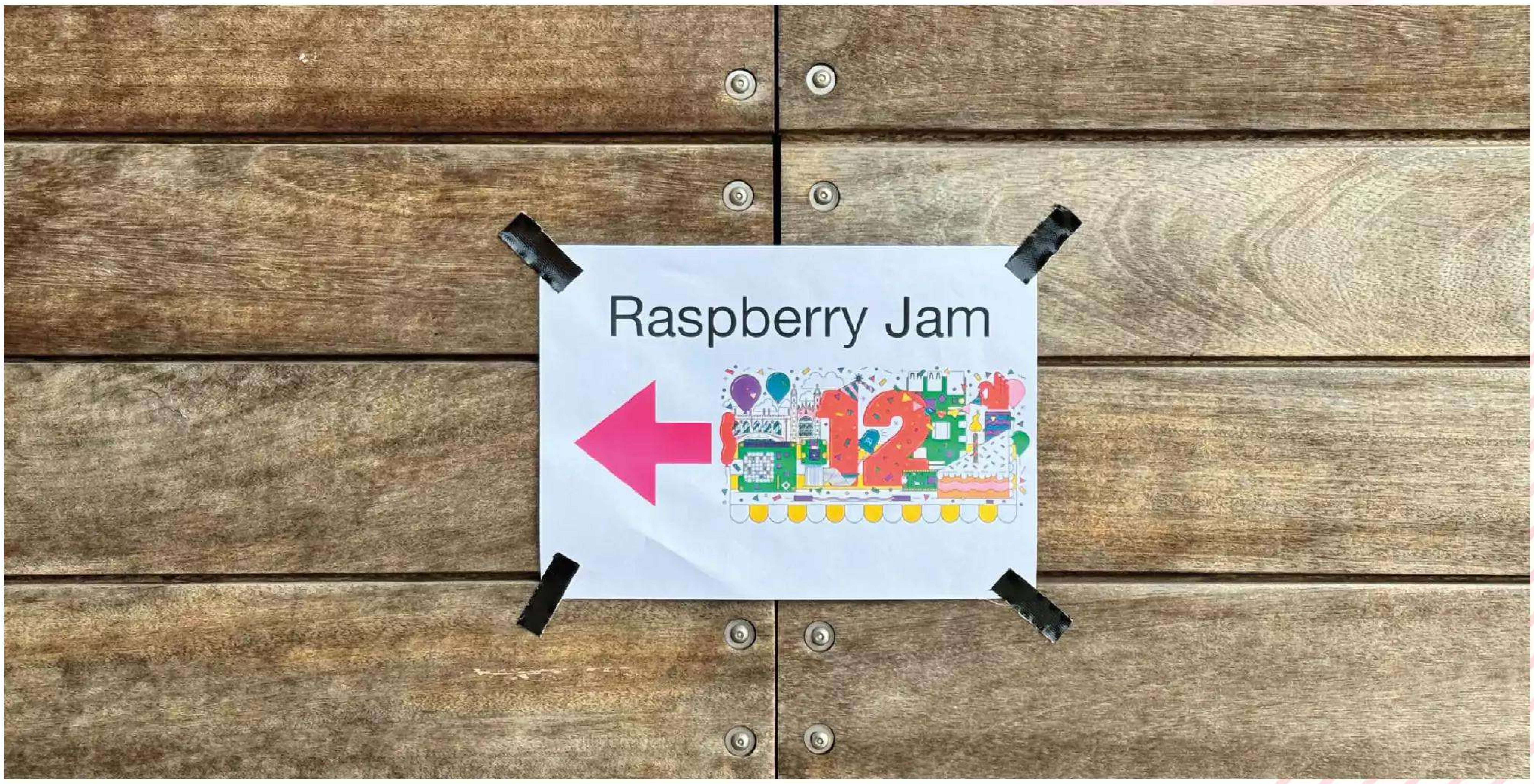
Recently I also started to explore working with ceramics, and how to bring out the personal artwork in such tactile material. M



▲ Some of the more recent, simple (and very stylish) graphic design that has been the work of Sara

Raspberry Pi is 12!

We celebrated Raspberry Pi's 12th (or maybe third?) birthday at CamJam



On February 29 2012, Raspberry Pi was released to world. Or at least, a few thousand were. Demand was phenomenal and the initial stock ran out very fast. The rest, as they say, is history.

It's been a long while since Raspberry Pi has had a proper chance to celebrate its birthday – releasing on a leap year day aside, the tenth anniversary happened a little too close to COVID for people to be comfortable attending big events, so it only seems right to make a bit of a deal of it this year.

CamJam is one of the oldest running Raspberry Jams, taking place in the city where Raspberry Pi was born it's also where the spin-off Pi Wars event was created, and has hosted Raspberry Pi on many occasion in the past. For this special CamJam, Features Ed Rob from *The MagPi*, along with our Publishing Director Brian Jepson, were there showing off our books and magazines alongside the folks from the Raspberry Pi Store selling other Raspberry Pi wares.

The turnout was fairly modest compared to Jams past, however it was a great time. Foundation folks attended to present coding workshops, and Eben Upton held a special anniversary talk and Q&A with Gordon Hollingworth (CTO) and James Adams (COO).

The community is still as strong as ever 12 years on. Here's to 12 years more.

01. The Computer Science Lab has been home to CamJam for a while now, and makes for a great venue

02. The ever popular Pi Noon was on display, as robots battled robots to pop each other's balloon

03. It's a lot cheaper than couples therapy

04. Talks were held throughout the day in the main lecture room for a small change of pace

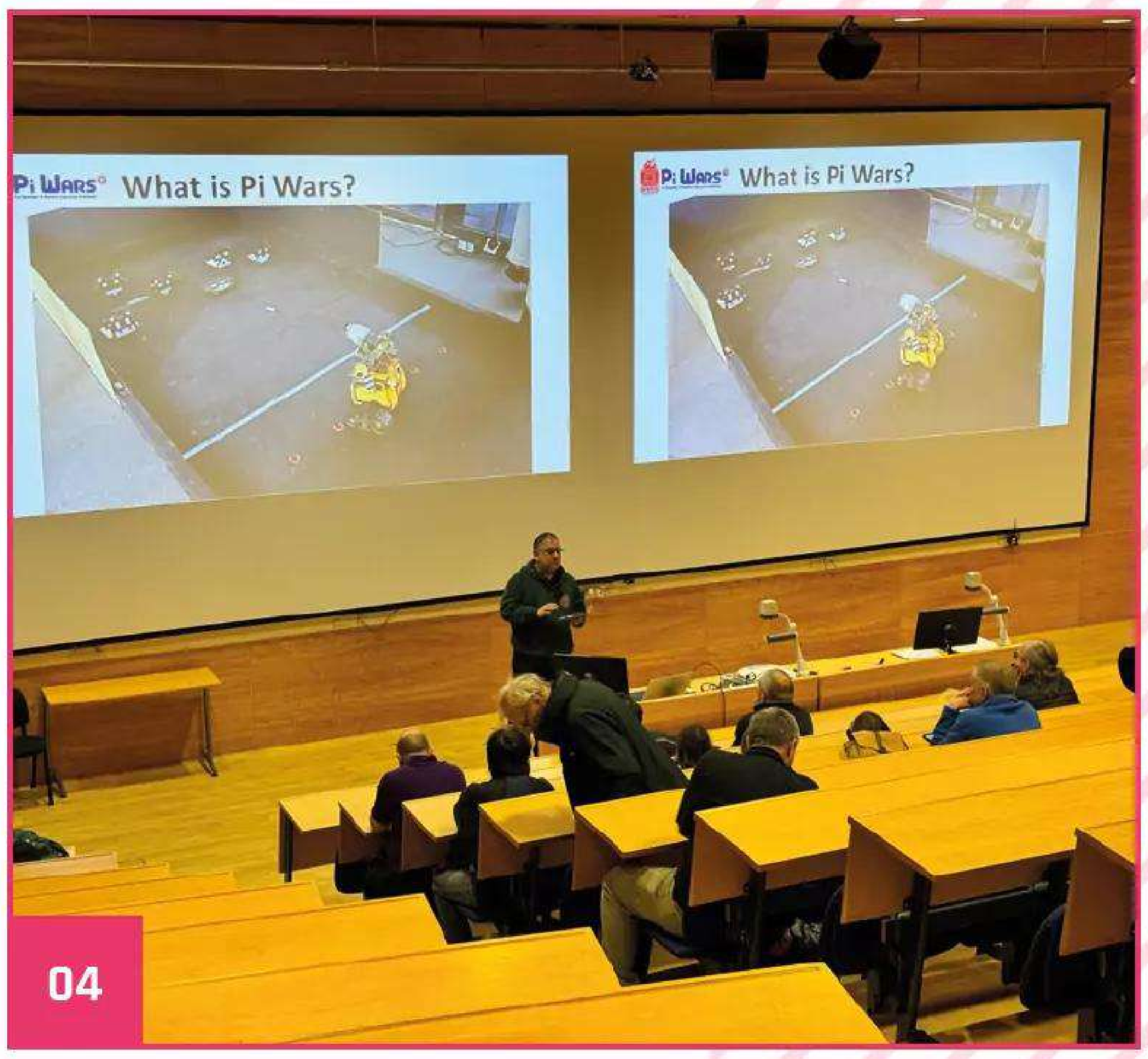
05. Kids love robots, it's a universal truth that is unlikely to change

06. Resident maker Toby Roberts had plenty of cool Raspberry Pi-powered projects on show

01

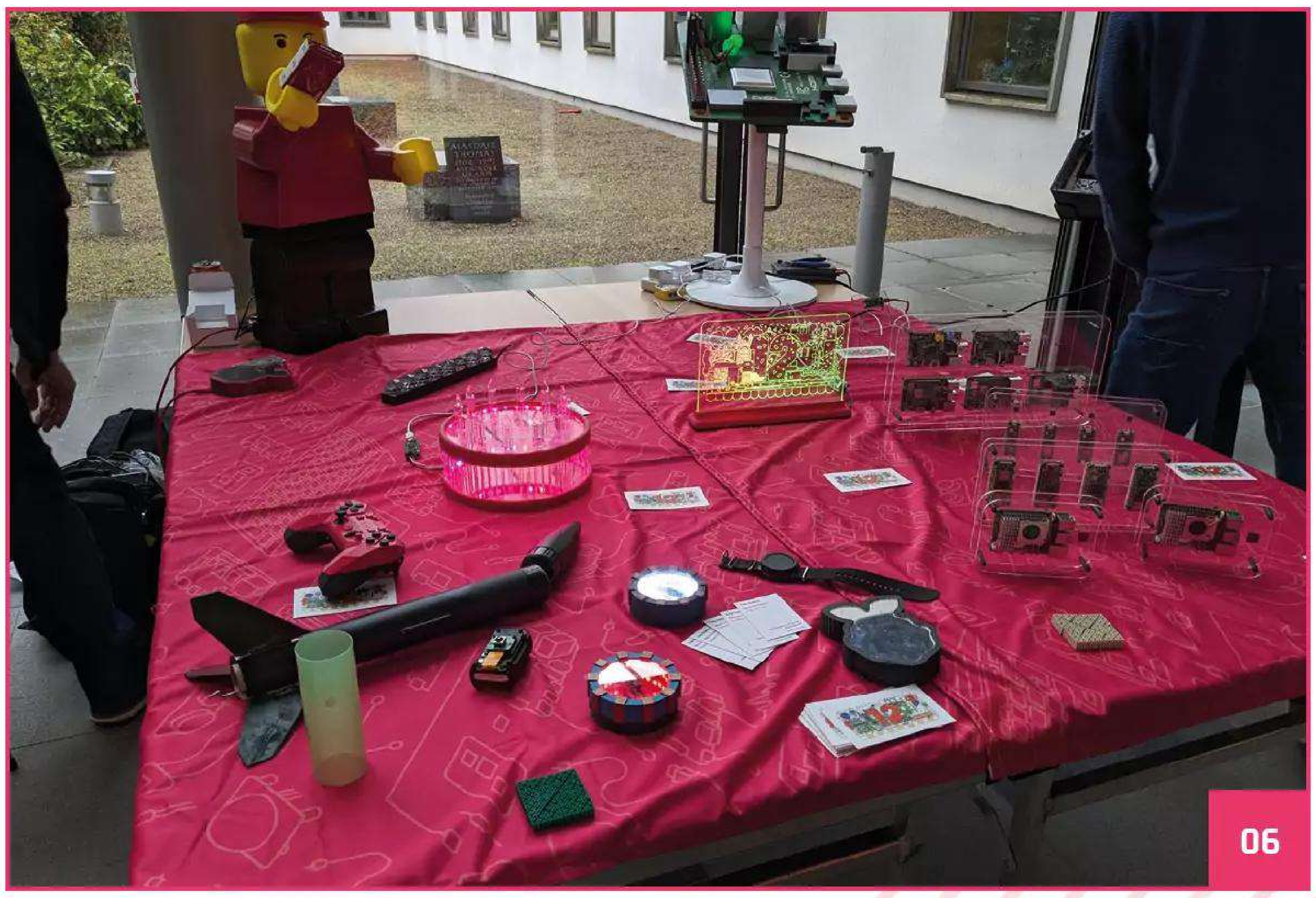
02

03



04

05



06

MagPi Monday

Amazing projects direct from social media!

Every Monday we ask the question: have you made something with a Raspberry Pi over the weekend? Every Monday, our followers send us amazing photos and videos of the things they've made.

Here's a selection of some of the awesome things we got sent this month. Remember to follow along at the hashtag #MagPiMonday!

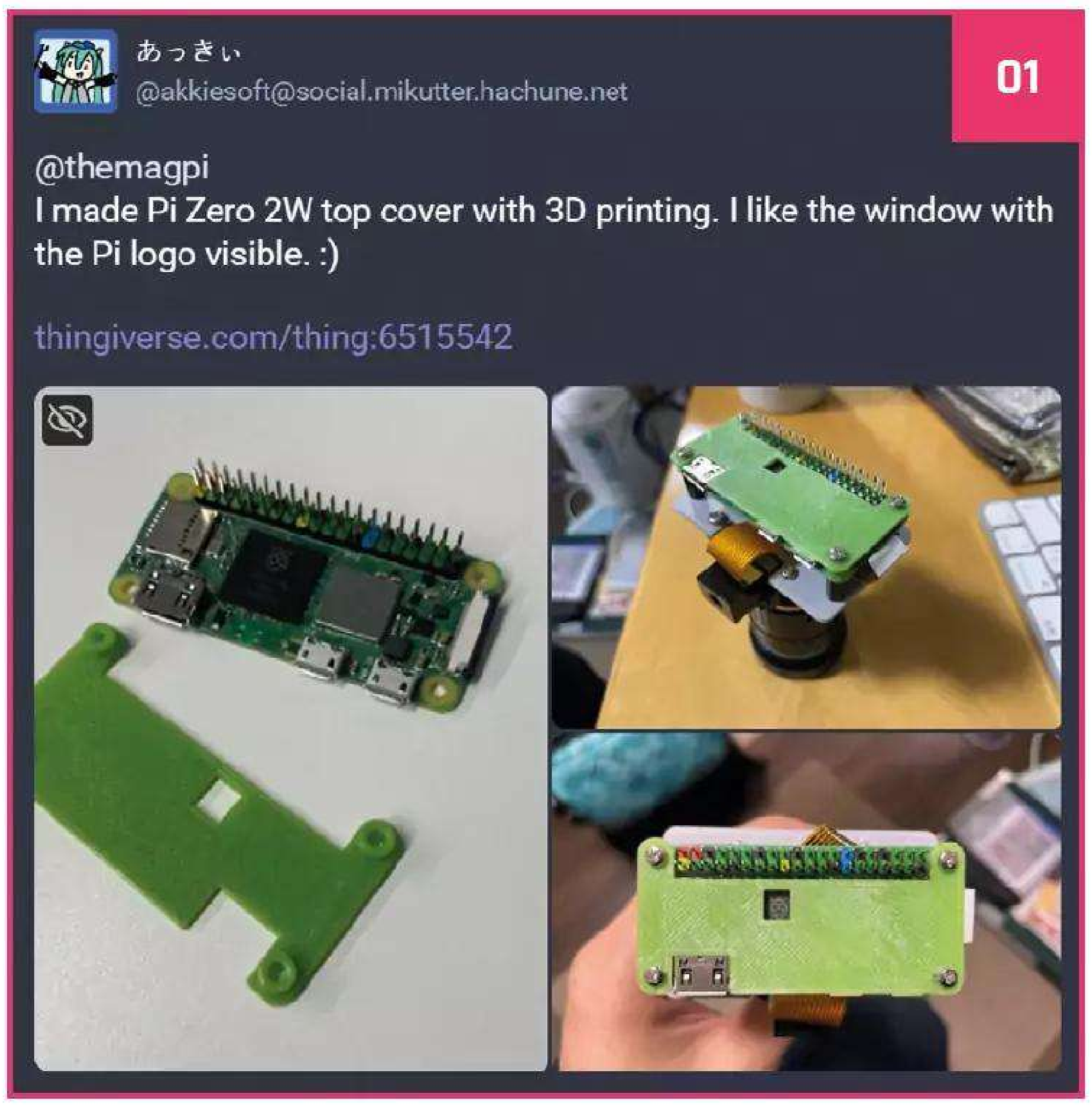
01. A great 3D printed case from the excellent Akkie! We like the window

02. Ooh, connecting two screens at once in such an efficient way is a great idea

03. You can make a robot chassis out of anything, including a bit of cardboard

04. A very handy little cal status thing! A great thing to put on your laptop

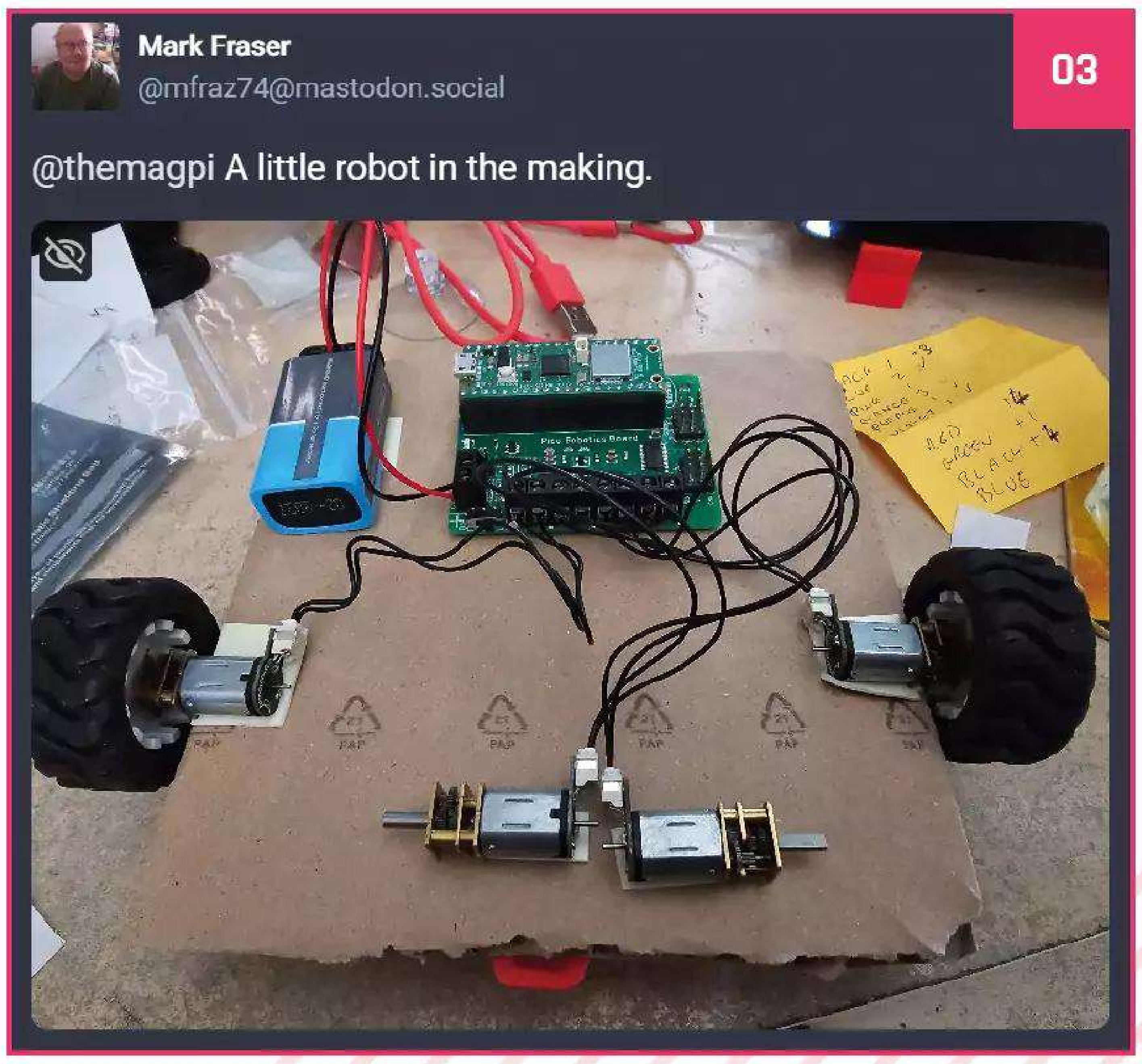
05. James of the Raspberry Jam Berlin has been working on very cool stuff with the new Picamera library


01

あっきぃ
@akkiesoft@social.mikutter.hachune.net

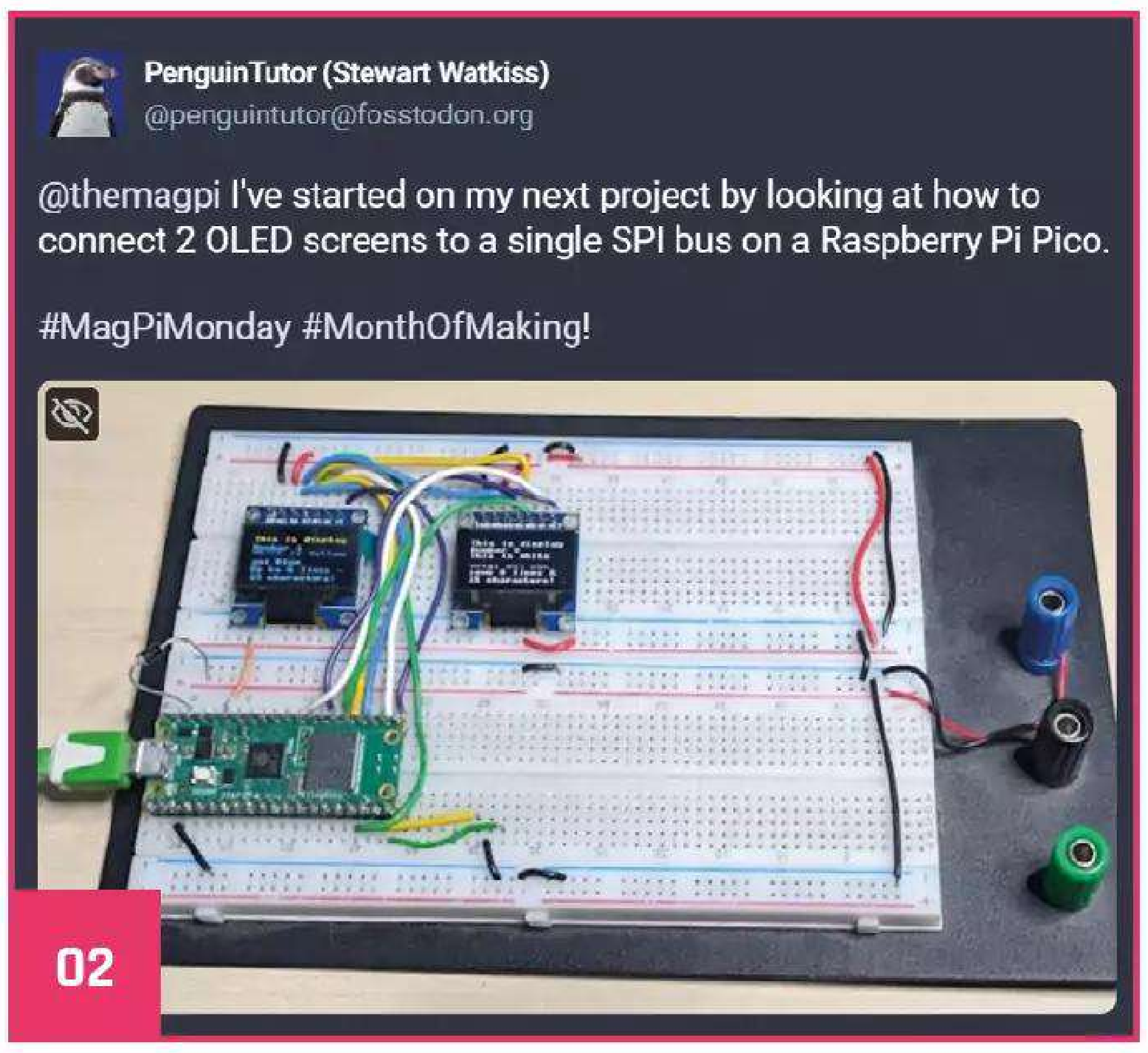
@themagpi
I made Pi Zero 2W top cover with 3D printing. I like the window with the Pi logo visible. :)

thingiverse.com/thing:6515542


03

Mark Fraser
@mfraz74@mastodon.social

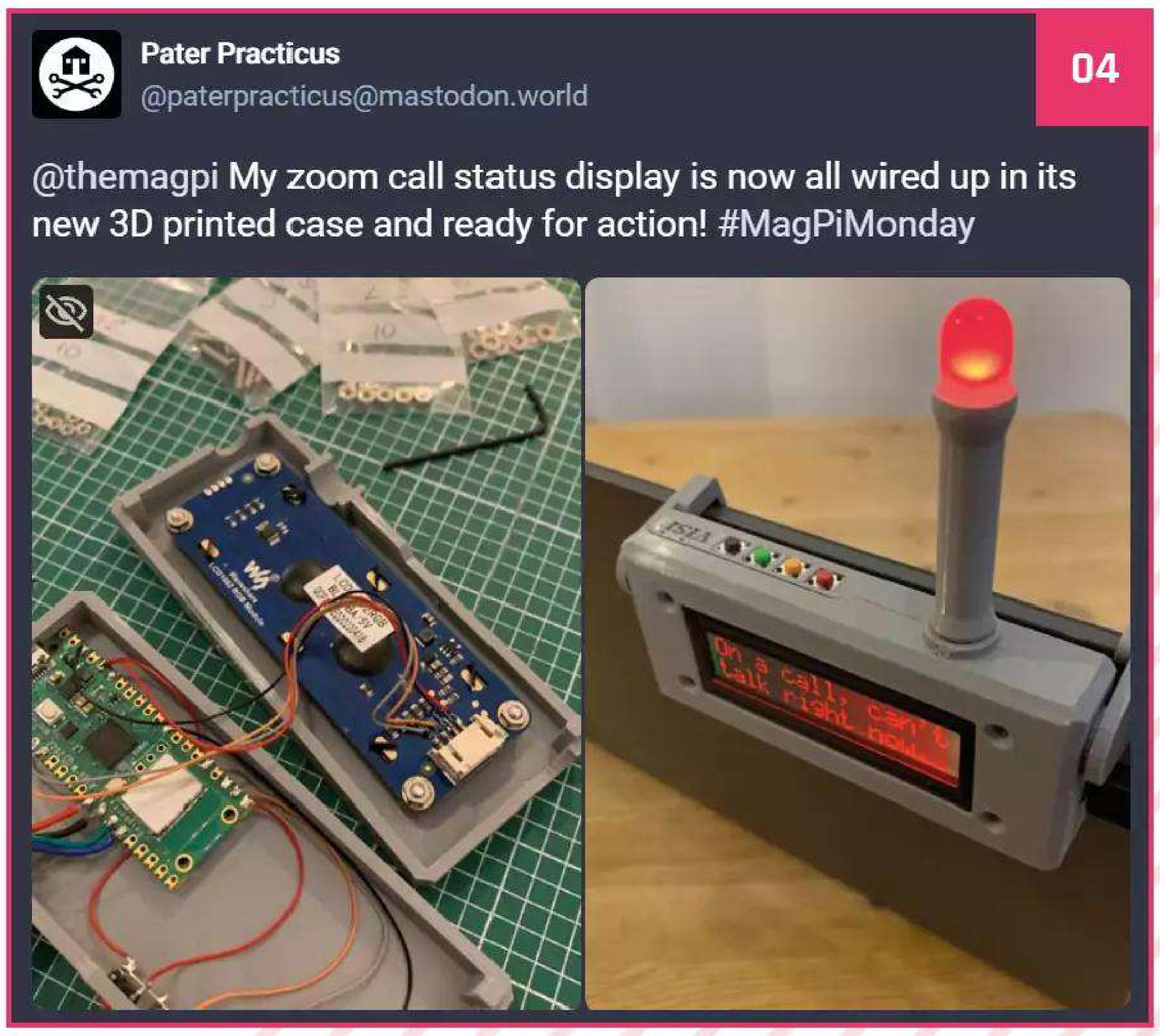
@themagpi A little robot in the making.


PenguinTutor (Stewart Watkiss)
@penguintutor@fosstodon.org

@themagpi I've started on my next project by looking at how to connect 2 OLED screens to a single SPI bus on a Raspberry Pi Pico.

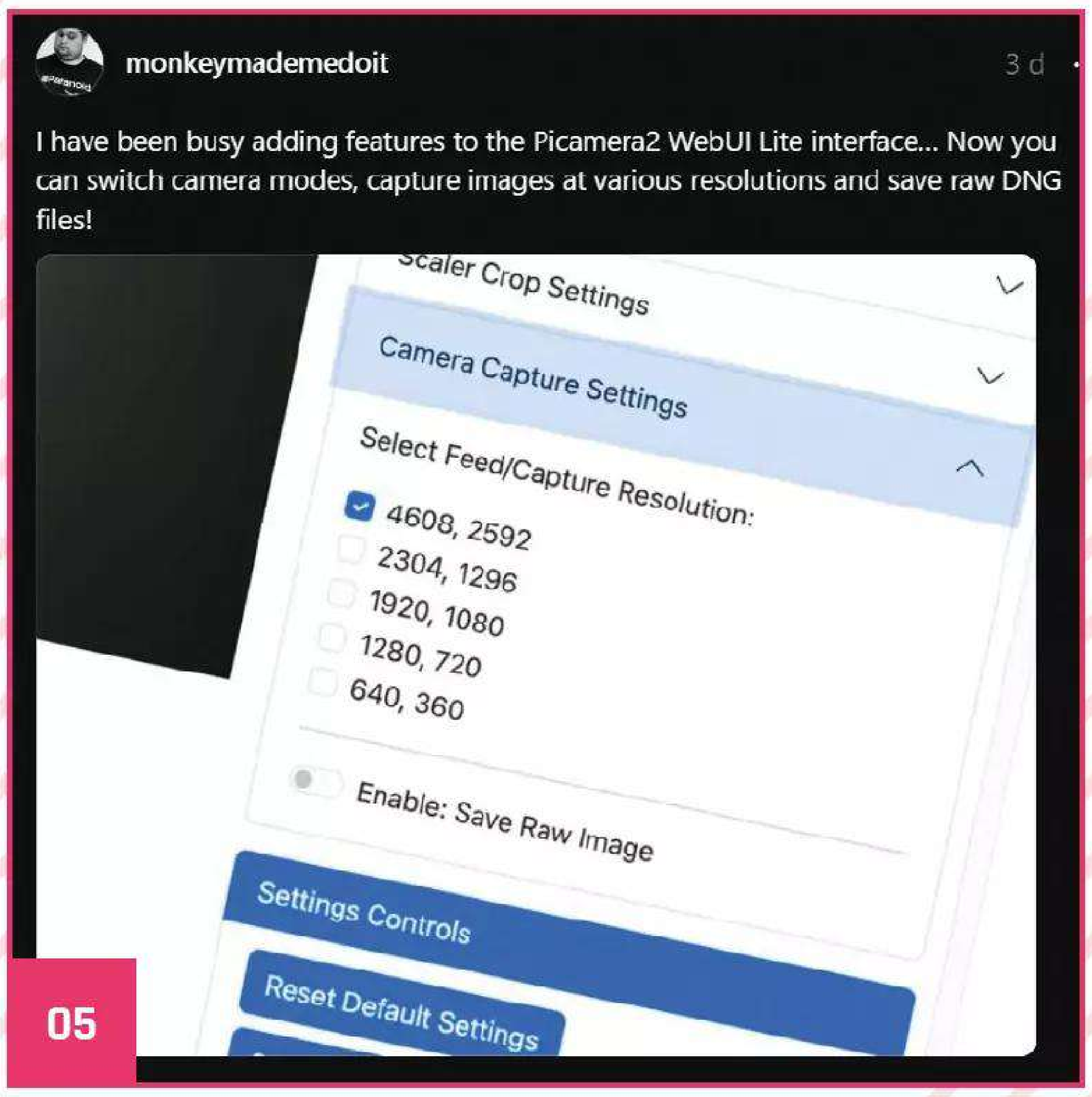
#MagPiMonday #MonthOfMaking!

02


04

Pater Practicus
@paterpracticus@mastodon.world

@themagpi My zoom call status display is now all wired up in its new 3D printed case and ready for action! #MagPiMonday


05

monkeymademedoit 3 d

I have been busy adding features to the Picamera2 WebUI Lite interface... Now you can switch camera modes, capture images at various resolutions and save raw DNG files!

Scaler Crop Settings

Camera Capture Settings

Select Feed/Capture Resolution:

☑ 4608, 2592
□ 2304, 1296
□ 1920, 1080
□ 1280, 720
□ 640, 360

○ Enable: Save Raw Image

Crowdfund this

Great crowdfunding projects this month

2x2 Square and Round Quad Display

2X2 SQUARE & ROUND
QUAD DISPLAY
Driven by Raspberry Pi Pico and ESP32 Powered Display consoles

Available with both round and square screens, this little board can be powered by a Raspberry Pi Pico to show multiple bits of data on one device. Usually you'd need to hook a load of these together, so it's very neat to see them on one board for a dashboard or other similar project.

▶ kck.st/3P1QifV

NVME Raspberry Pi 5 Case

The NVMe Base from Pimoroni is pretty popular – we reviewed it last issue and quite liked it – so it's no surprise to see people start making cases for Raspberry Pi 5 that will also fit the Base. This also has excellent ventilation and a little LCD screen for data on what we assume will be a NAS box for a lot of folks.

▶ kck.st/3SWTF93

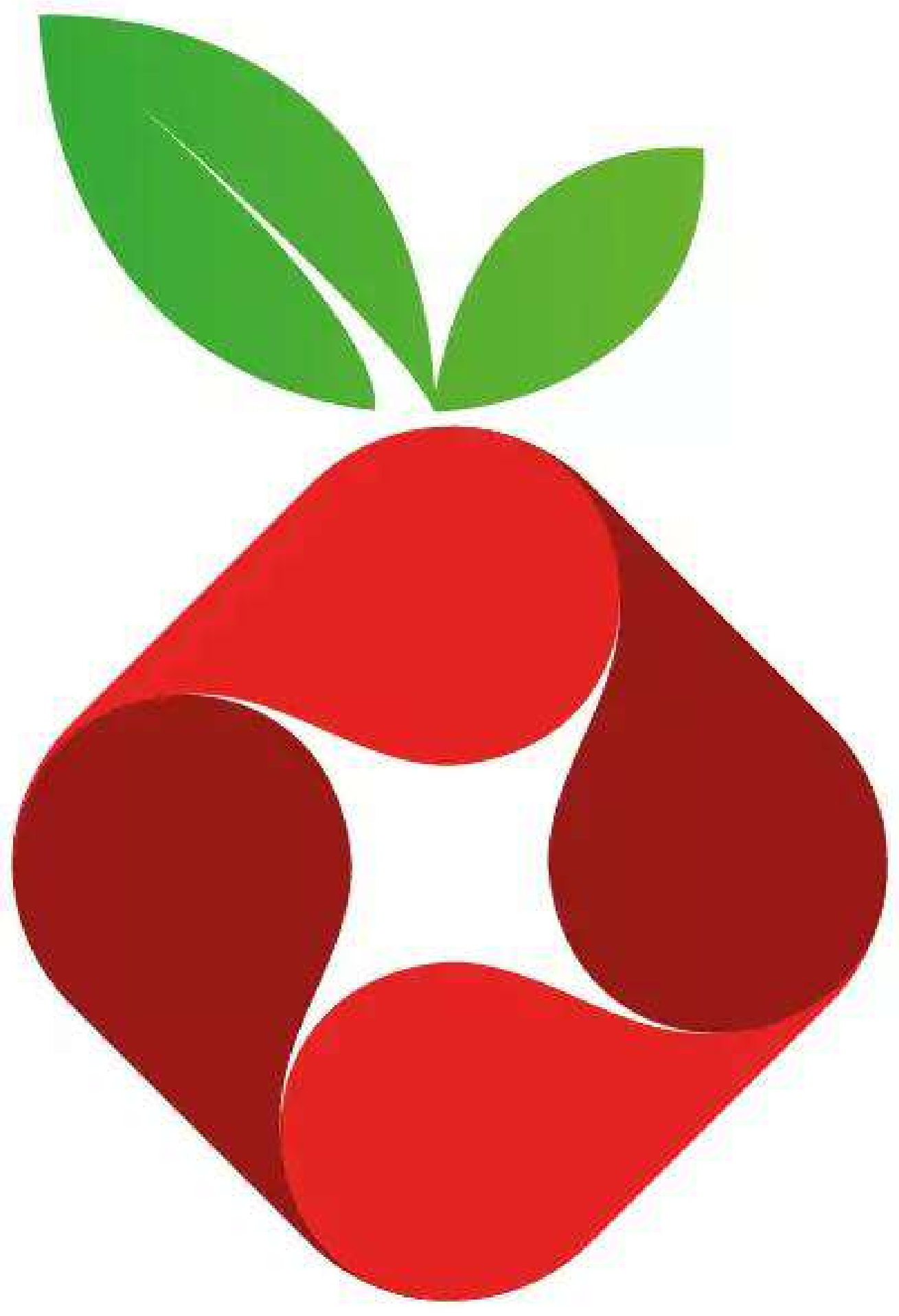
Your Letters

Security articles

I'm a subscriber to your magazine. Do you have a search tool for the articles and/or projects? I'm looking for firewall/security advice, as I've recently purchased a Raspberry Pi 5.

Mike via email

We currently don't have an official database of our articles for the magazine, although we know some readers have set some up in the past. However we don't really have many articles on Raspberry Pi software/network security – although we did cover setting up Pi-Hole in issue 104 (**magpi.cc/104**). We'll have a look to see if we can put together some tutorials on other router and firewall solutions using Raspberry Pi.



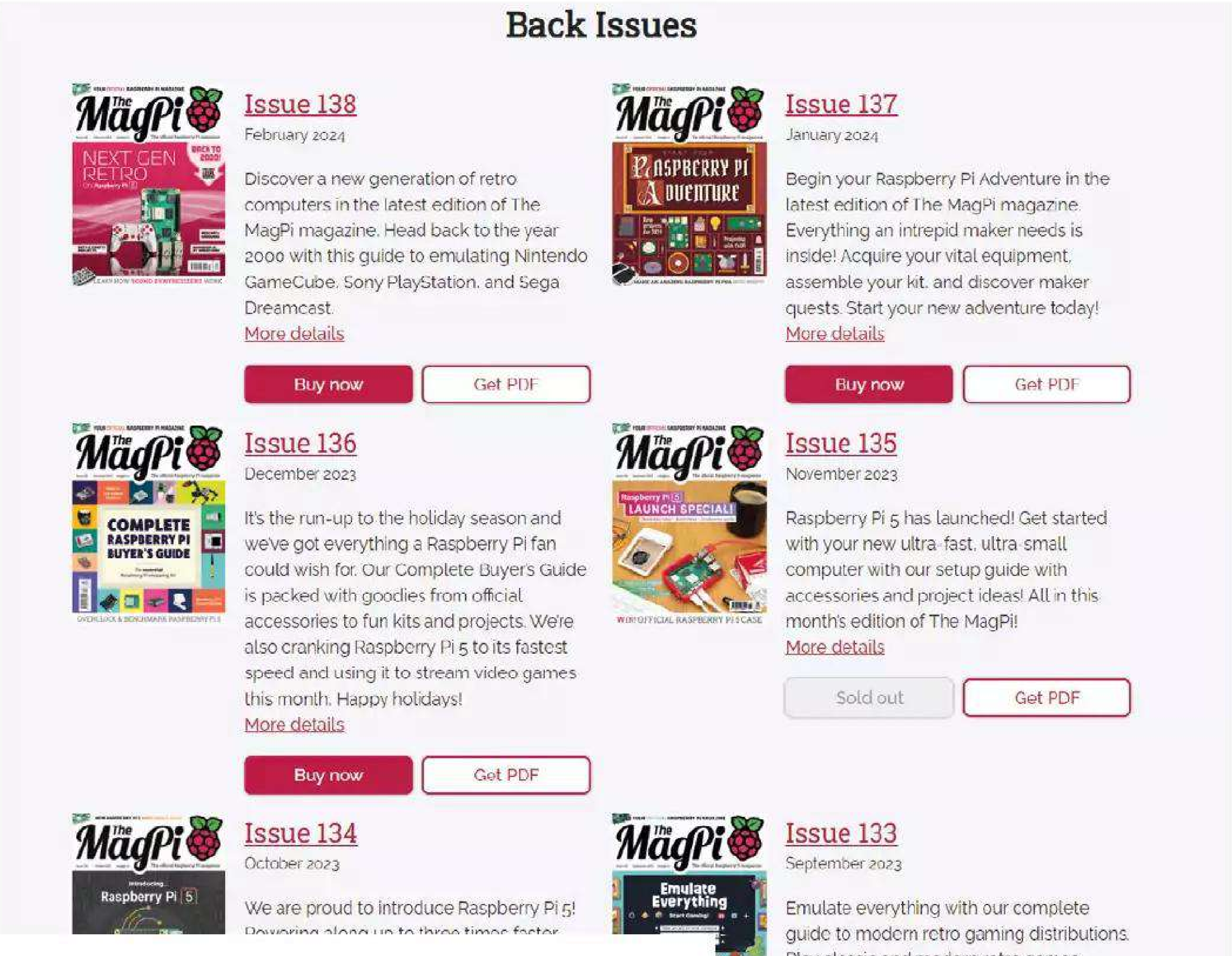
▲ With Pi-Hole you can introduce network-wide ad blocking, which is good first step in adding security to your home or office

Issues galore

I am rather new to the DIY and maker scene and was wondering if there is a way to download ALL back issues of The MagPi at one time?

Michael via email

There's no way to grab them all at once, however on Raspberry Pi OS if you head to the Bookshelf app you can download the last ten issues very quickly. If you head to **magpi.cc/issues**, it's also pretty quick to get the free PDFs. If you want more of the highlights of the magazine, check out our Official Projects Books and Official Handbooks at **magpi.cc/books**



Back Issues

Issue 138
February 2024

Discover a new generation of retro computers in the latest edition of The MagPi magazine. Head back to the year 2000 with this guide to emulating Nintendo GameCube, Sony PlayStation, and Sega Dreamcast.
More details

Issue 137
January 2024

Begin your Raspberry Pi Adventure in the latest edition of The MagPi magazine. Everything an intrepid maker needs is inside! Acquire your vital equipment, assemble your kit, and discover maker quests. Start your new adventure today!
More details

Issue 136
December 2023

It's the run-up to the holiday season and we've got everything a Raspberry Pi fan could wish for. Our Complete Buyer's Guide is packed with goodies from official accessories to fun kits and projects. We're also cranking Raspberry Pi 5 to its fastest speed and using it to stream video games this month. Happy holidays!
More details

Issue 135
November 2023

Raspberry Pi 5 has launched! Get started with your new ultra-fast, ultra-small computer with our setup guide with accessories and project ideas! All in this month's edition of The MagPi!
More details

Issue 134
October 2023

We are proud to introduce Raspberry Pi 5!
Powering along up to three times faster

Issue 133
September 2023

Emulate everything with our complete guide to modern retro gaming distributions. Play classic and modern retro games, emulate consoles and vintage desktop

▲ All of our back issues are available as free PDFs

Laser radar facts?

I enjoyed this article (and issue), but a couple of frustrating things.

The quick facts mention 3D printing at their local Tesco Express? Why no detail on this service, presumably referring to the screen enclosure?

Also it looks more like PCBs constructed into an enclosure for Raspberry Pi, definitely worth a mention.

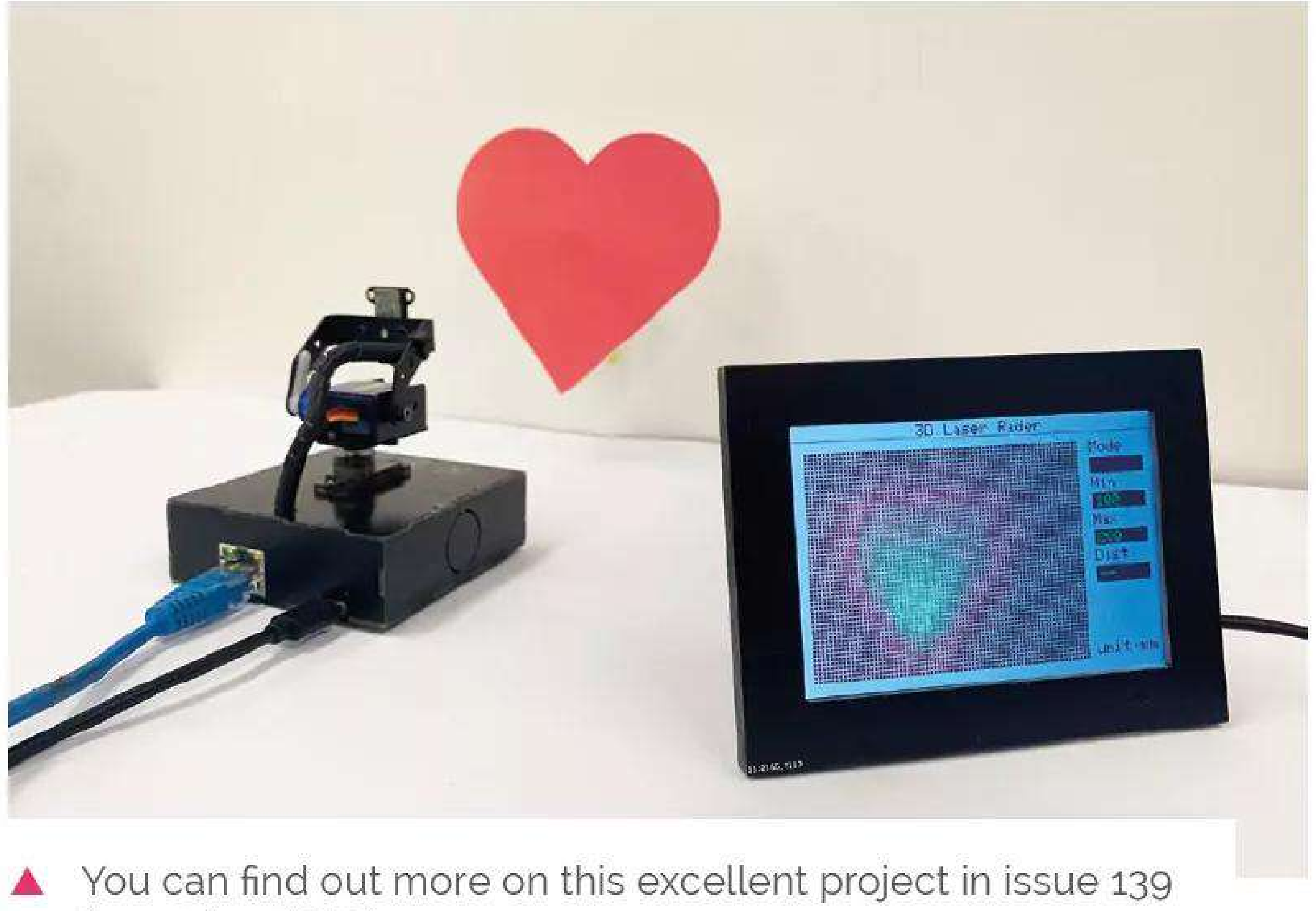
Lastly I think there needs to be better link checking as xyz.com doesn't exist.

Thanks for the continued production/hard work over many years, I always love it.

Tyeth via email

Ah… that is our mistake. What you've seen there is our placeholder text for the quick facts section, none of them are true or correspond to the project. Apologies that we didn't catch those. We wish you *could* 3D print stuff at Tesco Express!

For more info on the stuff maker Gavin Chang has created with similar PCBs (which explains the construction a little more) you can check his projects out here: **magpi.cc/gavinchang**



▲ You can find out more on this excellent project in issue 139 (**magpi.cc/139**)

Contact us!

- Mastodon **magpi.cc/mastodon**
- Threads **@themagpimag**
- Facebook **magpi.cc/facebook**
- Email **magpi@raspberrypi.com**
- Online **forums.raspberrypi.com**

USA SPECIAL!
6 ISSUES
FOR $43

YOUR OFFICIAL RASPBERRY PI MAGAZINE
The MagPi
TROUBLESHOOTING GUIDE
UPCYCLE OLD SONOS SYSTEMS WITH RASPBERRY PI

FREE
RASPBERRY PI
PICO W

Subscribe online:
magpi.cc/subscribe

Continuous credit card orders will auto-renew at the same price unless cancelled. A free Pico W is included with all subscriptions. This is a limited offer. Not included with renewals. Offer subject to change or withdrawal at any time.

Community Events Calendar

Find out what community-organised Raspberry Pi-themed events are happening near you…

01. Computer Vision-Based Electric Vehicle Charging Occupancy Detection

Tuesday 9 April

Messezentrum 1, Nürnberg, Germany

magpi.cc/cvbev140

As part of the 'Embedded Vision & Edge AI' track at Embedded World 2024, Arm's Sandeep Mistry is sharing a case study on computer vision-based electric vehicle charging occupancy detection, a project which uses Raspberry Pi Pico.

02. Easter 2024 Hythe CoderDojo

Thursday 11 April

Hythe Library, Southampton, United Kingdom

magpi.cc/ehcd140

We'll be exploring (subject to change): HTML/CSS, Scratch, Python, Astro Pi, Coolest Projects submissions, or bring your own project. At this event we are expecting Tom from CoderDojo to join us to award some prizes.

03. Raspberry Pi Jam Hyderabad

Saturday 20 April

Nirmal Vihar Apartment, Hyderabad, India

magpi.cc/rpjh140

Celebrating Raspberry Pi's birthday a bit late so as to accommodate for friends having final exams in school. This will ensure we have time to celebrate and learn new things for the summer vacation.

04. Pi Wars 2024

Saturday 20 April to Sunday 21 April

William Gates Building, Cambridge, UK

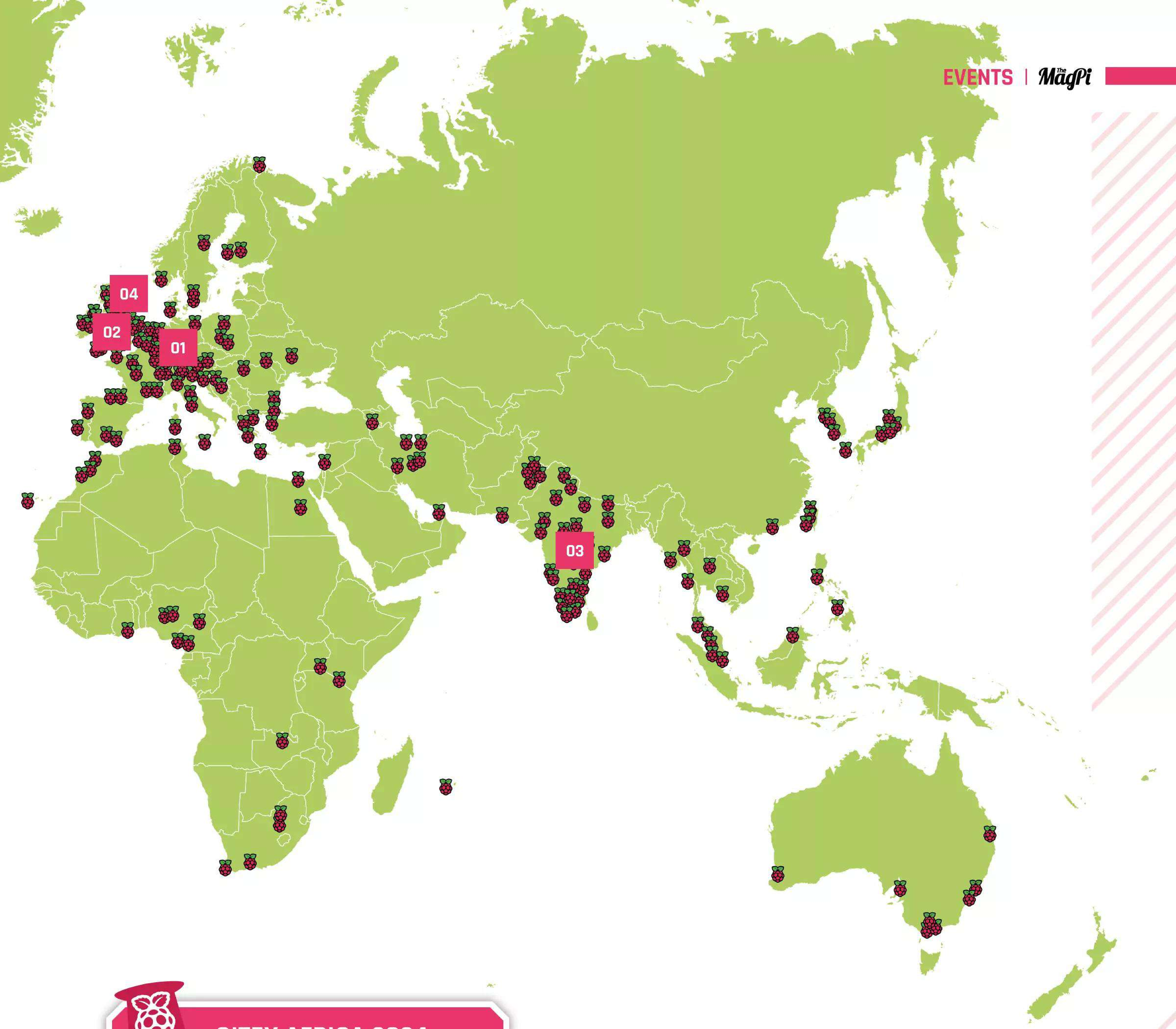
piwars.org

Pi Wars is a non-destructive Raspberry Pi-based robotics competition with both autonomous and remote-controlled challenges. It takes place over one weekend and features teams of school students, family members and hobbyists as well as solo roboteers. The event is open to anyone on planet Earth!

FULL CALENDAR

Get a full list of upcoming community events here:

magpi.cc/events



GITEX AFRICA 2024
MEET RASPBERRY PI

Gitex Africa 2024

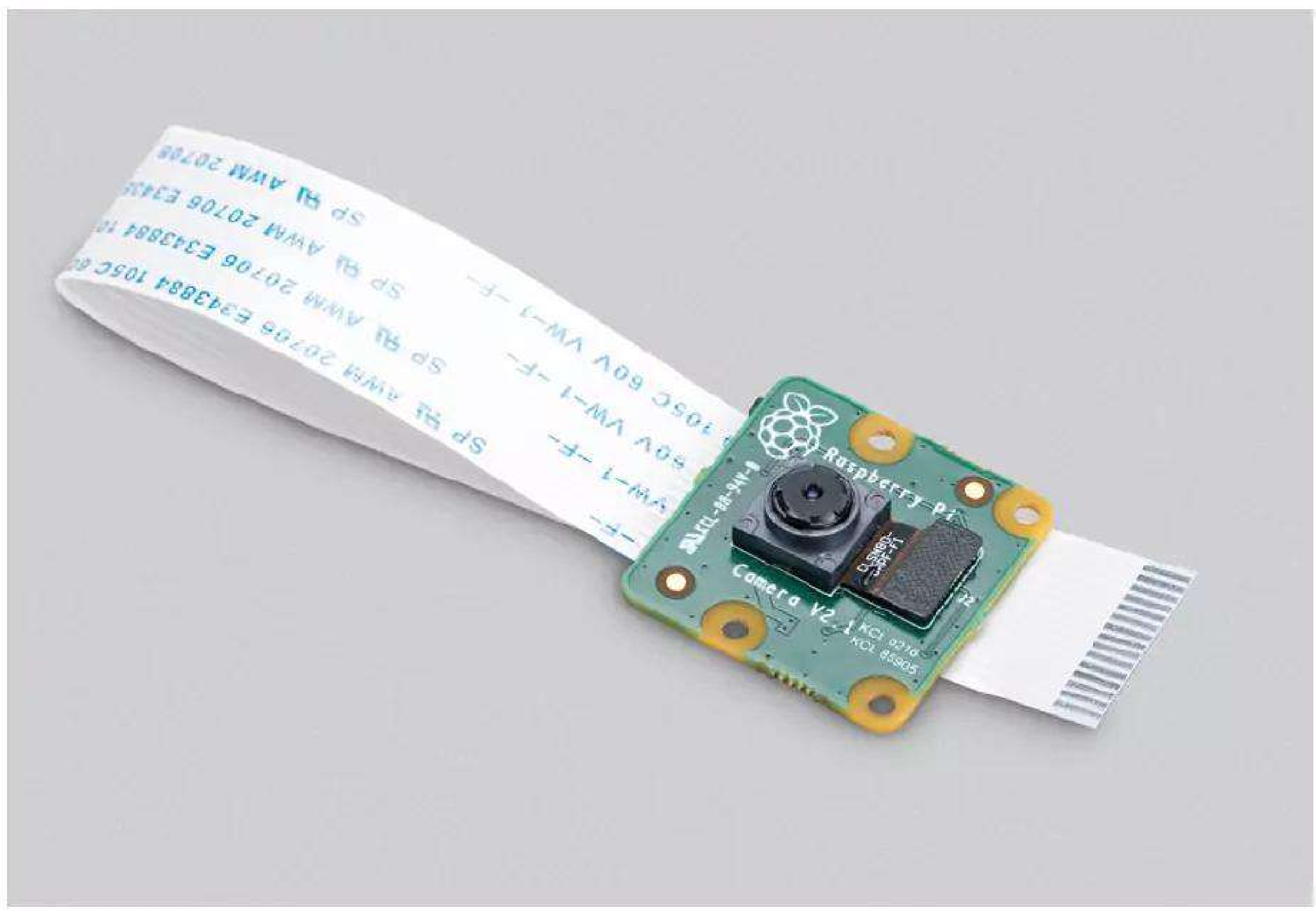
- Where Place Bab Jdid, Marrakesh, Morocco
- When Wednesday 29 May to Friday 31 May

The Raspberry Pi team is delighted to be in Marrakesh, Morocco for our first visit to Africa's biggest tech and startup show: Gitex Africa 2024. There you'll be able to meet our team and experience the full range of our technology, including Raspberry Pi Pico, RP2040-based solutions, Compute Modules, Raspberry Pi single-board computers as well as cameras and our range of high-quality accessories.

magpi.cc/gitex24

WIN ONE OF FIVE RASPBERRY JAM BUNDLES!

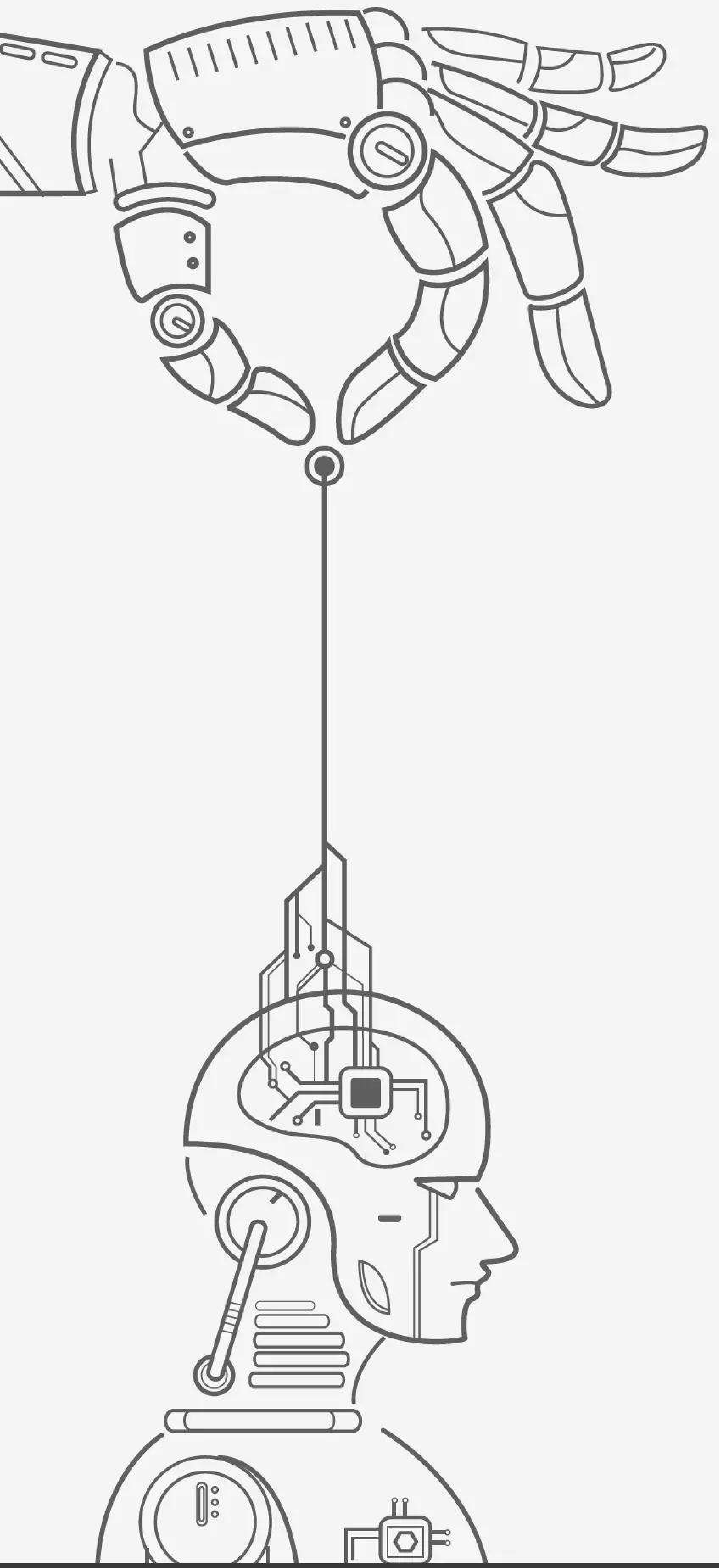
Did you know? If you run a Raspberry Pi community event, you get swag for your community. Over the months of April and May in 2024, a subscription to this very magazine is one of the items event runners will receive. As you're reading about this here, we hope you already have a subscription, so we thought we'd give you all a chance to win the rest of the kit's contents: a Raspberry Pi Zero 2 W and a Raspberry Pi Camera Module 2.



Head here to enter: **magpi.cc/win** | **Learn more:** **magpi.cc/events (or check out page 94!)**

Terms & Conditions

Competition opens on **27 March 2024** and closes on **25 April 2024**. Prize is offered to participants worldwide aged 13 or over, except employees of Raspberry Pi Ltd, the prize supplier, their families, or friends. Winners will be notified by email no more than 30 days after the competition closes. By entering the competition, the winner consents to any publicity generated from the competition, in print and online. Participants agree to receive occasional newsletters from *The MagPi* magazine. We don't like spam: participants' details will remain strictly confidential and won't be shared with third parties. Prizes are non-negotiable and no cash alternative will be offered. Winners will be contacted by email to arrange delivery. Any winners who have not responded 60 days after the initial email is sent will have their prize revoked. This promotion is in no way sponsored, endorsed or administered by, or associated with, Instagram, Facebook, X or any other companies used to promote the service.



Learn AI with Raspberry Pi

Your friendly introduction to artificial intelligence technology and tools

The Magpi **#141**
On sale **25 April**

Plus!

Raspberry Pi 5 cases on test!

Control industrial robot arms

Computing roots with Pi-PDP 10

DON'T MISS OUT! **magpi.cc/subscribe**

MASTODON	magpi.cc/mastodon
THREADS	@themagpimag
FACEBOOK	magpi.cc/facebook
EMAIL	magpi@raspberrypi.com
ONLINE	forums.raspberrypi.com

EDITORIAL

Editor
Lucy Hattersley
lucy@raspberrypi.com

Features Editor
Rob Zwetsloot
rob@raspberrypi.com

Sub Editor
Ian Evenden

ADVERTISING

Charlotte Milligan
charlotte.milligan@raspberrypi.com
+44 (0)7725 368887

DESIGN

Head of Design
Jack Willis

Designers
Sara Parodi, Natalie Turner

Illustrator
Sam Alder

Photographer
Brian O'Halloran

CONTRIBUTORS

David Crookes, PJ Evans, Gareth Halfacree, Rosie Hattersley, Phil King, Rob Miles, K.G. Orphanides

PUBLISHING

Publishing Director
Brian Jepson
brian.jepson@raspberrypi.com

Director of Communications
Liz Upton

CEO
Eben Upton

DISTRIBUTION

Seymour Distribution Ltd
2 East Poultry Ave,
London EC1A 9PT
+44 (0)207 429 4000

SUBSCRIPTIONS

Unit 6 The Enterprise Centre
Kelvin Lane, Manor Royal,
Crawley, West Sussex, RH10 9PE
+44 (0)1293 312193
magpi.cc/subscribe
magpi@subscriptionhelpline.co.uk

This magazine is printed on paper sourced from sustainable forests and the printer operates an environmental management system which has been assessed as conforming to ISO 14001.

The MagPi magazine is published by Raspberry Pi Ltd, 194 Cambridge Science Park, Milton Road, Cambridge, England, CB4 0AB. The publisher, editor, and contributors accept no responsibility in respect of any omissions or errors relating to goods, products, or services referred to or advertised in the magazine.

Except where otherwise noted, content in this magazine is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported (CC BY-NC-SA 3.0).

ISSN: 2051-9982.

Coding in an AI age

Why learn to code when ChatGPT can do it for you? By **Lucy Hattersley**

We live in what the Chinese reputedly referred to as 'interesting times'! LLMs (Large Language Models) such as GPT, LLama and MMLU have shaken the world, and the world of computing in particular.

It's worth remembering that the "T" in GPT stands for "transformer" (Generative Pre-Trained Transformer in full). GPT isn't creating its own words, it's transforming your input into a response through a highly educated guess based on its training model. It does not – yet – seem to understand what it's saying. And it's often misguided. But let's not dwell on that and look at the positives.

In the case of ChatGPT this training model is huge. GPT-4 is trained on 13 trillion tokens (parts of words) and around 10 trillion words. The training set clearly includes a lot of code from web sources, forums and books.

> "John von Neumann might disagree if he was around, but somehow I feel he'd be delighted"

Artists are understandably less than thrilled that AI is producing facsimiles of their work without giving them credit, or payment. For coders: AI changes everything. It can help you write, explain, understand, and improve the quality of code, and increase productivity by enhancing performance. It's versatile in all programming languages and can help translate code between them.

On the downside, GPT can spit out code that kind of works for people who sort of understand it. And, as it gets better, they may not need, or even want, to understand it.

Abstract arts

Everybody involved in technology knows about abstraction. The process whereby the intricate technology stack gets hidden away, and the user is presented with a simpler interface. The iPhone is easier than the GUI PC, which is easier than the DOS PC, which is easier than the PDP. This next step: the ChatGPT "How can I help today?" rather than an IDE and knowledge of coding.

Raspberry Pi exists, on some level, counter to abstraction. We want to tear people away from their shiny slabs of glass and glue, and show them the insides of a computer. "It isn't magic, it's just billions of on/off switches flicking on and off at a billion times per second!" Which is, in itself, a form of magic.

Anybody doubting the importance of GPT and similar technologies isn't really paying attention. The negative responses remind me of Douglas Adams' three rules:

1. Anything that is in the world when you're born is normal and ordinary and is just a natural part of the way the world works.
2. Anything that's invented between when you're 15 and 35 is new and exciting and revolutionary and you can probably get a career in it.
3. Anything invented after you're 35 is against the natural order of things."

Most of us are somewhere between two and three, but objections to AI aren't just Ludditism. When applied to creative arts, AI devalues human involvement and can be accused of plagiarism. The same can be said of code, of course, but art feels instinctively more personal.

Coding is an incredibly cerebral process and requires creativity and deep thought. But coders stand on the shoulders of giants. I may understand a merge-sort algorithm, but I sure as heck didn't come up with it. And using AI to put it to work and explain it to me feels inherently useful. John von Neumann might disagree if he was around, but somehow I feel he'd be delighted.

AUTHOR

Lucy Hattersley

Lucy is editor of *The MagPi* and didn't find a way to shoe-horn a reference to The Culture in this month, but go and read Iain M Banks anyway.

magpi.cc

HiPi.io

HIGHPI PRO

The new case from the HiPi.io team

- Rapid tool-free assembly and disassembly
- Large internal volume for HATs
- Compatible with Pi 2/3/4
- Multiple lid options
- Passive & Active Cooling options
- Secure microSD card cover
- VESA mount support
- Molding-configurable output ports customizable for volume orders
- Printed logo for your branding

Available at these great Pi stores:

PiShop.us PiShop.ca PiShop.mx ThePiHut Pi-Shop.ch Welectron.

Contact your favorite Pi store if it's not listed here

PiKVM

Remote control redefined

Manage your servers or PCs remotely!

POWERED BY Raspberry Pi

PiKVM V4 Mini

Small, cost-effective, and powerful!

- Power consumption in idle mode: just 2.67 Watts!
- Transfer your mouse and keyboard actions
- Access to all configuration settings like UEFI/BIOS
- Capture video signal up to 1920x1200@60 Hz
- Take full control of a remote PC's power

PiKVM V4 Plus

The most feature-rich edition

- More connectivity
- Extra storage via internal USB 3.0
- Upgraded powering options
- More physical security features
- Extra HDMI output
- Advanced cooling solution

A cost-effective solution for data-centers, IT departments or remote machines!

Available at the main Raspberry Pi resellers

core ELECTRONICS
PiShop.ca
RPishop.cz
Welectron.
elektorstore
PIMORONI
Pi-Shop.ch
The PiHut
PiShop.us
CloudFree
KUBii
Cytron Marketplace
ROBU.IN Your ideas, Our Parts
NETTOP COMPUTERS & ELECTRONICS
netsket mall
DIREKTRONIK
RaspberryPi.dk

HiPi.io

No reseller in your country?
Check shop.hipi.io (import fees might apply).

List of official
resellers by country: